P. 1877/83

1 1983

informatyka

W NUMERZE:

TECHNIKA MIKROPROCESOROWA

WYDAWNICTWO SIGMA

Przedsmak mikro-rewolucji

Najnowszy Raport dla Klubu Rzymskiego, noszący tytuł: „Mikroelektronika i społeczeństwo. Na dobre i złe", przynosi wizję gigantycznej eksplozji mikroelektroniki i możliwych — związanych z nią — społecznych perturbacji. Mimo stosunkowo sporej wiedzy o trwającym w krajach wysoko rozwiniętych mikroprocesorowym boomie, trudno nam pojąć ogrom płynących z niego ułatwień i zagrożeń. Mikroinformatyczne szaleństwo mamy bez wątpienia daleko przed sobą — w odległej, niełatwej do wyobrażenia perspektywie.

Jeszcze informatyka nie zawładnęła polskimi umysłami, jeszcze nie czuje się jej dyktatu, ale już z pesymistycznych nawet wyliczeń wynika, że firmie polonijnej opłaca się wyprodukować dla kraju dość tani komputer osobisty, że to czy tamto przedsiębiorstwo chętnie użyje nawet niedoskonałych nowych zdobyczy informatyki, że mikroukłady opłaca się sprowadzać za czarnorynkowe dolary... Nasza mamutowata gospodarka z niechęcią odwraca się od nowinek informatycznych, ale już na jej kuchennych schodach widać pierwszych emisariuszy rewolucji.

Rozwój techniki mikroprocesorowej w Polsce jest powolny, dokonywany jakby na przekór wszystkiemu, ale — wyraźny. Począwszy od niniejszego numeru, postaramy się na naszych łamach aktywniej wspierać ten rozwój. Informatyka nie jest bowiem w stanie (ze względu na bardzo ograniczone możliwości techniczne) zwiększyć męczącego ciśnienia struktur biurokratycznych na społeczeństwo, na poszczególne jednostki, a przez to — pogłębić alienacji i frustracji Polaków. Świadomie rozwijana mogłaby się natomiast przyczynić do złagodzenia naszych codziennych trudności.

Nie znaczy to jednak, że problemy społeczeństw rozwiniętych gospodarczo mamy pozostawić im samym, by za 10 czy 15 lat — sami dotknięci bakcylem nowych technologii — gorączkowo przypominać sobie dzisiejszą lekcję z Zachodu. Dlatego też za kilka miesięcy przedstawimy fragmenty Raportu „Mikroelektronika i społeczeństwo". Ku zachęcie i przestrodze.

REDAKCJA

NACZELNA ORGANIZACJA TECHNICZNA

WYDAWNICTWO SIGMA

CZASOPISM I KSIĄŻEK TECHNICZNYCH

ul. Świętokrzyska 14a
00-950 Warszawa
skrytka pocztowa 1004

KOLEGIUM REDAKCYJNE

Redaktor naczelny: prof. dr hab. Leon ŁUKASZEWICZ

mgr inż. Zbigniew GLUZA, dr Janusz GWIAZDA, Władysław KLEPACZ (zastępca redaktora naczelnego), dr inż. Tomasz PAWLAK, mgr Andrzej SZAŁAS, mgr Zbigniew ŚWIRSKI, dr inż. Janusz ZALEWSKI

Sekretarz redakcji: mgr Teresa JABŁOŃSKA

RADA PROGRAMOWA

Prof. dr hab. Tadeusz PECHE (przewodniczący), mgr inż. Tomasz BAŃKOWSKI (sekretarz), mgr inż. Antoni BOSSOWSKI, mgr inż. Roman BURNO, prof. dr hab. Andrzej JANICKI, mgr inż. Jan KRAMARCZUK, prof. dr hab. inż. Juliusz KULIKOWSKI, prof. dr hab. Leon ŁUKASZEWICZ, prof. dr hab. Antoni MAZURKIEWICZ, gen. dr inż. Marian PASTERNAK, dr inż. Bronisław PIWOWAR, mgr Zbigniew SUBSTYK, prof. dr hab. Tadeusz WALCZAK

Materiałów nie zamówionych Redakcja nie zwraca.

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 129 i 133, tel. 27-71-40, dyżury redakcji 10.00—12.00

Zakł. Graf. „Tamka". Zam. 333. Obj. 5,5 ark. druk. Nakład 3700 egz. M-72.

Cena egzemplarza zł 75.— INDEKS 36124 Prenumerata roczna zł 900.—

P 1877/83

Plusczok K.: RTDS-8 — system wspomagający uruchamianie systemów mikroprocesorowych

INFORMATYKA 1983, nr 1, s. 5

Charakterystyka systemu RTDS-8, opracowanego w Zakładzie Systemów Automatyki Kompleksowej PAN w Gliwicach. Opisano konfigurację sprzętu i oprogramowanie, a także uniwersalny emulator mikroprocesorów 8-bitowych. W zakończeniu podano zamierzenia związane z dalszym rozwojem tego systemu oraz fakt rozpoczęcia produkcji w Zakładach Urządzeń Komputerowych MERA-ELZAB w Zabrzu.

Плусчок К.: RTDS-8 — вспомогательная система пускания в действие микропроцессоровых систем

INFORMATYKA 1983, № 1, стр. 5

Характеристика системы RTDS-8, разработанной Отделением Системов Комплексной Автоматики ПАН в Гливицах. Описано конфигурацию оборудования и программное обеспечение, а также универсальный эмулятор микропроцессоров 8-битовых. В заключении указано планы связанные с дальнейшим развитием этой системы и факт приступления к производству в Предприятии оборудований вычислительных машин MERA-ELZAB в Забжу.

Sinkiewicz T.: Mikroprocesorowy system wspomagania projektowania

INFORMATYKA 1983, nr 1, s. 8

Charakterystyka uniwersalnego systemu wspomagania MSWP przeznaczonego do projektowania, uruchamiania i testowania systemów mikroprocesorowych. Omówiono konfigurację sprzętu oraz oprogramowanie zbadane podczas eksploatacji egzemplarzy serii informacyjnej MSWP.

Синкевич Т.: Микропроцессоровая система вспомоществования проектирования

INFORMATYKA 1983, № 1, стр. 8

Характеристика универсальной системы вспомоществования MSWP предназначенной для проектирования, введения в действие и испытывания программ микропроцессоровых систем. Обсуждено конфигурацию оборудования и программное обеспечение, исследованное во время эксплуатации экземпляров информационной серии MSWP.

Pluta A., Poznański Z., Szynka J.: Wybrane systemy wspomagania dla mikrokomputerów rodziny MCS-48

INFORMATYKA 1983, nr 1, s. 10

Charakterystyka najczęściej stosowanych systemów wspomagania mikrokomputerów jednoukładowych. Omówiono metodykę oraz cztery różne rozwiązania systemów wspomagania, ilustrowane porównaniem ich możliwości funkcjonalnych.

Плута А., Познаньски З., Шынка Й.: Избранные системы вспомоществования для микро-ЭВМ семьи MCS-48

INFORMATYKA 1983, № 1, стр. 10

Характеристика наиболее часто применяемых систем вспомоществования односхемных микро-ЭВМ. Обсуждено методику и четыре разных решения систем вспомоществования, поясняемых сравнением их функциональных возможностей.

Górnicki T., Szeżyńska M.: Środki wspomagające przygotowywanie i uruchomianie mikrokomputerów jednoukładowych MCS-48

INFORMATYKA 1983, nr 1, s. 13

Ogólna charakterystyka mikroprocesorowych systemów uruchomieniowych z podaniem przykładu realizacji tego rodzaju systemu w Instytucie Podstaw Elektroniki Politechniki Warszawskiej w oparciu o procesor INTEL 8080. Rozwiązanie zostało sprawdzone na przykładzie oprogramowania autonomicznego monitora graficznego oraz specjalizowanego kalkulatora do obliczania wyników testów psychologicznych.

Гурницки Т., Шежиньска М.: Средства вспомоществующие подготовку и наладку односхемных микро-ЭВМ MCS-48

INFORMATYKA 1983, № 1, стр. 13

Общая характеристика микропроцессоровых наладочных систем исходя из примера реализации такой же системы в Институте Основ Электроники Варшавского Политехнического Университета, основаясь на процессор INTEL 8080. Решение было проверено на примере программного обеспечения автономного графического монитора и специализованного калькулятора для вычисления итогов психологических тестов.

Jankowski M. T.: KRABUS — sprzęg wewnątrzkasetowy dla systemów mikrokomputerowych

INFORMATYKA 1983, nr 1, s. 17

Charakterystyka międzynarodowej i krajowej sytuacji w zakresie ujednolicenia sprzęgów wewnątrzkasetowych dla systemów mikrokomputerowych. Omówiono propozycje sprzęgu wewnątrzkasetowego KRABUS prowadzącą do rozwiązania zbieżnego z aktualnymi tendencjami światowymi.

Янковски М. Т.: KRABUS — интерфейс для систем микро-ЭВМ

INFORMATYKA 1983, № 1, стр. 17

Характеристика международного и отечественного положения в сфере унифицирования интерфейсов для систем микро-ЭВМ. Обсуждено предложение интерфейса KRABUS, ведущего к решению сходному с актуальными мировыми тенденциями.

Dańda J.: Oprogramowanie systemów mikroprocesorowych — praktyczne aspekty polityki producenta półprzewodników

INFORMATYKA 1983, nr 1, s. 20

Charakterystyka działań zmierzających do obniżenia społecznych kosztów tworzenia oprogramowania dla systemów mikroprocesorowych. Działania te realizowane przez Instytut Technologii Elektronowej w Warszawie polegały na opracowaniu metody programowania zwanej modelowaniem strukturalizowanym oraz uniwersalnej biblioteki programów, a także prowadzeniu szkolenia użytkowników.

Даньда Й.: Программное обеспечение микропроцессоровых систем — практические аспекты политики производителя полупроводников

INFORMATYKA 1983, № 1, стр. 20

Характеристика действий имеющих в виду снижение общественных расходов создавания программного обеспечания для микропроцессоровых систем. Деятельность эта, реализованная Институтом Электронной Технологии в Варшаве, заключалась в разработке метода программирования, называемого структурализованным моделированием и универсальной библиотеки программ, а также подготовка пользователей.

Kreczmar A., Salwicki A.: Język programowania LOGLAN. Część 3.

INFORMATYKA 1983, nr 1, s. 22

Ostatnia część charakterystyki języka LOGLAN zawierająca opis mechanizmów ochrony, sposobu realizacji obliczeń równoległych, programowanego „odśmiecania" oraz innych cech języka. W zakończeniu porównano LOGLAN z innymi językami programowania oraz omówiono perspektywy jego dalszego rozwoju.

Кречмар А., Сальвицки А.: Язык программирования LOGLAN. Часть 3

INFORMATYKA 1983, № 1, стр. 22

Последняя часть характеристики языка LOGLAN, содержающая описание механизмов защиты, способа реализации параллельных вычислений, программного „очищания" и других черт языка. В заключении сравнено LOGLAN с другими языками программирования и обсуждено перспективы его дальнейшего развития.

egz. 8104

Plusczok K.: RTDS — a system for development of microcomputers

INFORMATYKA 1983, No. 1, p. 5

Characteristics of the RTDS system, elaborated in the Laboratory for Integrated Automation of Polish Academy of Science in Gliwice. Hardware configuration and software, as well as general purpose emulator for 8-bit microprocessor, are discussed. In the conclusion plans for future development of the system, as well as information about production starting in MERA-ELZAB Computer Devices Works in Zabrze, are presented.

Plusczok K.: RTDS-8 — ein Mikroprozessorentwicklungssystem

INFORMATYKA 1983, Nr. 1, S. 5

Eine Charakteristik des RTDS-8 Systems, das im Anstalt für Komplexe Automatiksysteme der Polnischen Akademie der Wissenschaften erarbeitet wurde. Es wurden die Hardwarekonfiguration und Software, sowie ein universeller Emulator für 8-bit-Prozessoren beschrieben. Im Abschluss wurden die Pläne für die Weiterentwicklung des Systems, sowie die Tatsache der Herstellungsbeginn in Rechnerapparatur Werken MERA-ELZAB in Zabrze, angegeben.

Sinkiewicz T.: Microprocessor system for assisted designing

INFORMATYKA 1983, No. 1, p. 8

Characteristics of the MSWP general purpose system for microprocessor systems development. Hardware configuration and software, investigated during operation of MSWP information series specimens, are discussed.

Sinkiewicz T.: Mikroprozessorsystem für unterstützte Projektierung

INFORMATYKA 1983, Nr. 1, S. 8

Eine Charakteristik des universellen Unterstützungssystems, MSWP genannt, für Projektierung, Bau und Prüfung der Mikroprozessorsysteme. Es wurden Hardwarekonfiguration und Software, die während des Betriebes der Exemplare von MSWP Informationsserie getestet waren, besprochen.

Pluta A., Poznański Z., Szynka J.: Selected development systems for MCS-48 microcomputer family

INFORMATYKA 1983, No. 1, p. 10

Characteristics of the most often applied development systmes for single-chip microcomputers. Methodology and four different solutions of development systems, illustrated with comparison of their functional possibilities, are discussed.

Pluta A., Poznański Z., Szynka J.: Ausgewählte Entwicklungssysteme für MCS-48 Mikrorechnerfamilie

INFORMATYKA 1983, Nr. 1, S. 10

Eine Charakteristik der am meisten angewendeten Entwicklungssysteme für die Einzelschaltkreismikrorechner. Es wurde Methodik und vier verschiedene Lösungen der Entwicklungssysteme, illustriert mit einem Vergleich ihrer Funktionsmöglichkeiten, besprochen.

Górnicki T., Szeżyńska M.: Media for development of MCS-48 single-chip microcomputers

INFORMATYKA 1983, No. 1, p. 13

General characteristics of microprocessor development systems with realization example of similar type of system based on INTEL 8080 processor in the Electronics Foundations Institute at Warsaw Technical University. The solution was verified during elaboration of autonomy software for graphical display and specialized calculator for counting results of psychological tests.

Górnicki T., Szeżyńska M.: Die Hilfsmittel für Projektierung und Entwicklung der MCS-48 Einzelschaltkreismikrorechner

INFORMATYKA 1983, Nr. 1, S. 13

Eine allgemeine Charakteristik der Mikroprozessorentwicklungssysteme mit einem Realisationsbeispiel von derartigem, auf Basis des INTEL 8080 gebauten, Systems im Institut für Elektronikgrundlagen von Warschauer Technischen Universität. Die Lösung wurde als autonomische Software des graphischen Bildschirms und des Spezialrechners für Ergebnissenberechnung der psychologischen Teste, geprüft.

Jankowski M. T.: KRABUS — interface for microcomputer systems

INFORMATYKA 1983, No. 1, p. 17

Characteristics of international and home situation in unificating of interfaces for microcomputer systems. A proposal for the interface KRABUS, which leads to convergent solution with actual world trends, is discussed.

Jankowski M. T.: KRABUS — eine Schnittstelle für Mikrorechnersysteme

INFORMATYKA 1983, Nr. 1, S. 17

Eine Charakteristik der internationalen und einheimischen Situation im Bereich der Schnittstellennormierung für Mikrorechnersysteme. Es wurde ein Vorschlag für die KRABUS-Schnittstelle, die alle aktuelle Welttendenzen umfasst, besprochen.

Dańda J.: Software for microprocessor systems — practical aspects of semiconductor manufacturer's policy

INFORMATYKA 1983, No. 1, p. 20

Characteristics of activities, which tends towards reduction of microprocessor system software costs. The activities, realized by the Electronic Technology Institute in Warsaw, consist in elaboration of a programming method, called „structuralized modelling", and universal program library, as well as in users education.

Dańda J.: Software für Mikroprozessorsysteme — die praktischen Aspekte der Halbleiterherstellerspolitik

INFORMATYKA 1983, Nr. 1, S. 20

Eine Charakteristik der Massnamen zur Kostensenkung im Bereich der Softwareerarbeitung für Mikroprozessorsysteme. Diese Massnamen, realisiert im Institut für Elektronische Technologie in Warschau, beruhen auf Bearbeitung einer Programmiermethode, genannt „strukturalisierte Modellieren" und einer universellen Programmbibliothek, sowie auf Anwenderschulung.

Kreczmar A., Salwicki A.: Programming language LOGLAN. Part 3

INFORMATYKA 1983, No. 1, p. 22

Last part of the LOGLAN programming language characteristics, which includes description of protection mechanisms, realization methods for parallel computing, programable garbage-clearing and other language features. In the conclusion, LOGLAN against other programming languages is compared and prospects of its future development are discussed.

Kreczmar A., Salwicki A.: Programmiersprache LOGLAN. Teil 3

INFORMATYKA 1983, Nr. 1, S. 22

Letzter Teil der LOGLAN-Sprache Charakteristik, die eine Beschreibung der Schutzmechanismen, Realisationsweise für Parallelrechnung, programmierten „Müllentfernung" und anderen Sprachmerkmale umfasst. Im Abschluss wurde LOGLAN mit anderen Programmiersprachen verglichen und die Aussichten für seine Weiterentwicklung besprochen.

Informatyka

zastosowania w gospodarce, technice i nauce

Nr 1
MIESIĘCZNIK
1983
ROK XVIII
Styczeń

ORGAN KOMITETU INFORMATYKI, MINISTERSTWA NAUKI, SZKOLNICTWA WYŻSZEGO I TECHNIKI ORAZ KOMITETU NAUKOWO-TECHNICZNEGO NOT DS. INFORMATYKI

W NUMERZE: Strona

TECHNIKA MIKROPROCESOROWA — jest jak jest
Janusz Zalewski 4

RTDS-8 — system wspomagający uruchamianie systemów mikroprocesorowych
Krzysztof Plusczok 5

Mikroprocesorowy system wspomagania projektowania
Tadeusz Sinkiewicz 8

Wybrane systemy wspomagania dla mikrokomputerów rodziny MCS-48
Adam Pluta, Zbigniew Poznański, Jerzy Szynka 10

Środki wspomagające przygotowywanie i uruchamianie oprogramowania mikrokomputerów jednoukładowych MCS-48
Tadeusz Górnicki, Magdalena Szeżyńska 13

KRABUS — sprzęg wewnątrzkasetowy dla systemów mikrokomputerowych
Marek Tadeusz Jankowski 17

Oprogramowanie systemów mikroprocesorowych — praktyczne aspekty polityki producenta półprzewodników
Jerzy Dańda 20

Język programowania LOGLAN. Część 3
Antoni Kreczmar, Andrzej Salwicki 22

Zwiększenie efektywności mikrokomputera
oprac. Maria Waśniewska 26

ALGORYTMY

Problem FIND-UNION
Andrzej Szałas, Zbigniew Świrski 27

Zmodyfikowane sortowanie bąbelkowe
Andrzej Szałas, Zbigniew Świrski 29

Z KRAJU

Zastosowanie systemów informatycznych w szkołach wyższych
Elżbieta Kierczuk, Bogusław Żygadło 30

GIEŁDA INFORMACJI 32—34

ZE ŚWIATA

SICOB'82
Oprac. Marianna Sobczyk 32

Convention Informatique
Michał Ziębiński 34

RECENZJE

Próba czasu
Leonid Bułhak 36

Trzy słowniki
Adam B. Empacher 37

TERMINOLOGIA

Słowniczek mikroprocesorowy (cz. 1)
Marek Tadeusz Jankowski 38

GRY

Polowanie na UFO
Z. Świrski 40

POGLĄDY

Ceny i monopol
Janusz Gwiazda III okł.

Technika mikroprocesorowa — jest jak jest

W jakim stadium rozwoju jest technika mikroprocesorowa w Polsce? Czy pełna odpowiedź jest możliwa? Należałoby w niej uwzględnić wiele aspektów i przedstawić wiele powiązań zewnętrznych. Jeżeli świadomie ograniczymy się do zagadnień wyłącznie technicznych, to — według tradycyjnego podziału — trzeba zapytać o stan produkcji sprzętu, oprogramowania i zastosowania. Biorąc jednak pod uwagę dziejową rolę tej techniki, należy przede wszystkim uwzględnić rozwój zastosowań mikroprocesorów w gotowych wyrobach i systemach. Do jej stosowania niezbędny jest zaś sprzęt, który to umożliwi. Wynika stąd następne pytanie: c z y m i z c z e g o r o b i ć.

Odpowiedź na pytanie „z czego robić" przedstawimy w następnym numerze, który będzie zawierał opis szeregu konstrukcji mikroprocesorowych. Pytanie „czym robić" jest pytaniem o narzędzia — od oscyloskopu i analizatora, przez zestawy uruchomieniowe i systemy wspomagania, do oprogramowania skrośnego. Z tej gamy niezbędnych narzędzi o różnym stopniu złożoności — od czysto układowych, do czysto programowych — w bieżącym numerze przedstawiamy cztery systemy wspomagania (ang. development systems).

Uzasadnienie konieczności posiadania systemu wspomagania jest banalne, ale nie wszyscy są świadomi, że „goły" mikroprocesor niewiele różni się od tranzystora — pozostaje układem elektronicznym. Próba zbudowania mikrokomputera spełniającego założone funkcje może — bez tych narzędzi — okazać się chybiona. System wspomagania konieczny jest do rozpoczęcia prac zastosowaniowych. Jest to narzędzie kosztowne, ale płaci się za nie, by ułatwić i przyspieszyć właściwe prace, a więc — przy umiejętnym wykorzystaniu — nakłady mogą zwrócić się wielokrotnie.

Przy okazji warto zaznaczyć, że system wspomagania nie może być celem ostatecznym. Proces uruchamiania jest jedynie fragmentem całego cyklu życia mikrokomputera, ogniwem w procesie tworzenia produktu, zatem system wspomagania jest tylko jednym z niezbędnych narzędzi i wcale nie najbardziej wszechstronnym. Czy systemy wspomagania w obecnej postaci sprostają np. zadaniom specyfikacji i projektowania zestawów mikrokomputerowych, choćby — przygotowania dokumentacji? Chyba nie, ze względu na ograniczone możliwości.

Uruchamianie mikrokomputera spełniającego określone funkcje jest obecnie długotrwałe i wymaga wielu iteracji, polegających na usuwaniu błędów. Dzieje się tak dlatego, że błędy są konsekwencją niedoskonałego przygotowania projektu na etapach poprzedzających uruchamianie, np. niekompletnej lub nieformalnej specyfikacji wymagań. Wraz z rozwojem metod inżynierii oprogramowania (ang. software engineering) uruchamianie i wspomaganie projektowania mikrokomputerów nie będzie rozumiane tak wąsko jak dotychczas. Jednym ze sposobów doskonalenia klasycznych systemów wspomagania jest zintegrowanie ich z systemami oprogramowania skrośnego, np. w naszych warunkach — opracowanymi na komputerach serii SM, ODRA, MERA lub RIAD. Choć oprogramowanie skrośne jest bardziej wydajne niż systemy wspomagania, nie może być jedyne, gdyż nie spełnia ich podstawowej funkcji, tzn. nie zapewnia integracji sprzętu i oprogramowania.

Oprócz artykułów dotyczących krajowych systemów wspomagania, w bieżącym numerze przedstawiamy projekt standardowego sprzęgu mikrokomputerowego KRABUS oraz metodę tworzenia oprogramowania systemów mikrokomputerowych, nazwaną m o d e l o w a n i e m s t r u k t u r a l i z o w a n y m. Pominiemy natomiast systemy wspomagania przeznaczone do projektowania mikrokomputerów w oparciu o mikroprocesory segmentowe, a także zagadnienia oprogramowania skrośnego, chociaż odpowiednie prace są prowadzone.

W odniesieniu do magistrali KRABUS oczekujemy reakcji Czytelników i dalszych propozycji specjalistów. Przedstawiona koncepcja jest konkretna i ciekawa, ale z pewnością nie zadowoli wszystkich. Oprogramowanie jest najbardziej krytycznym elementem zastosowania mikroprocesora, gdyż „napełnia go inteligencją", w martwy przedmiot „tchnie życie". Dlatego należy z uwagą śledzić rozwój prac krajowych w tym kierunku.

Zostawiając szczegóły pod rozwagę Czytelnikom, należy stwierdzić, że przydatności tych rozwiązań nie można jeszcze oceniać, pokaże to czas — lepsze wyprą gorsze. Normalizacja magistrali (w oparciu o KRABUS lub inne rozwiązania) jest konieczna, podobnie jak zaangażowanie większej liczby osób i pieniędzy w poważną produkcję oprogramowania (np. w oparciu o modelowanie strukturalizowane). Należy ubolewać, że perspektywy produkcji systemów wspomagania są bardzo mgliste. Dopiero po zakupieniu i sprawdzeniu pierwszych egzemplarzy będzie można coś powiedzieć na temat ich funkcjonalności czy niezawodności.

Reasumując — jedna propozycja magistrali standardowej, anemiczne prace nad rozwojem oprogramowania nie czynią wiosny. Brak produkcji systemów wspomagania powoduje, że stan rozwoju techniki mikroprocesorowej można opisać lakonicznym — jest jak jest. Nie wygląda to optymistycznie w sytuacji, gdy — jak ktoś określił — nadszedł już czas, kiedy w tej grze nie będzie się liczył partner bez karty atutowej. Tą kartą w rozwoju zastosowań techniki mikroprocesorowej są systemy wspomagania, a w rozwoju gospodarczym kraju — mikroelektronika.

JANUSZ ZALEWSKI

KRZYSZTOF PLUSCZOK
Zakład Systemów Automatyki Kompleksowej PAN
Gliwice

RTDS-8 — system wspomagający uruchamianie systemów mikroprocesorowych

Wprowadzenie układów o dużym stopniu scalenia, głównie — mikroprocesorów i współpracujących z nimi układów programowanych, stworzyło nowe możliwości zastosowań techniki komputerowej i spowodowało powstanie szeregu nieznanych dotąd problemów. Wyłoniło się m.in. zagadnienie uruchamiania i testowania zestawów aplikacyjnych zawierających te złożone układy.

Do projektowania i uruchamiania systemów mikroprocesorowych zaczęto używać specjalizowanych narzędzi sprzętowych i programowych, zwanych s y s t e m a m i w s p o m a g a n i a. Środki wspomagające powinny spełniać szereg wymagań [5], związanych z:

— projektowaniem, uruchamianiem i testowaniem sprzętowej części systemu mikroprocesorowego

— projektowaniem, uruchamianiem, testowaniem i weryfikacją oprogramowania

— integracją sprzętu i oprogramowania.

Stopień spełnienia tych wymagań może stanowić kryterium podziału systemów wspomagania na trzy podstawowe klasy [5]:

- proste zestawy mikrokomputerowe
- systemy do opracowywania oprogramowania i prowadzenia prac modelowych przy użyciu systemu wspomagającego
- systemy wspomagające tworzenie oprogramowania oraz integrację sprzętu i oprogramowania w oparciu o emulację układową.

Systemy wspomagające trzeciej klasy umożliwiają integrację sprzętu i oprogramowania bezpośrednio przy użyciu systemu uruchamianego. Zazwyczaj zawierają one:

— mikrokomputer z pamięcią stałą i pamięcią operacyjną zależną od konfiguracji (tzn. rozszerzalną)

— rozbudowany system wejścia-wyjścia

— pamięć masową na dyskach elastycznych (z dyskowym systemem operacyjnym).

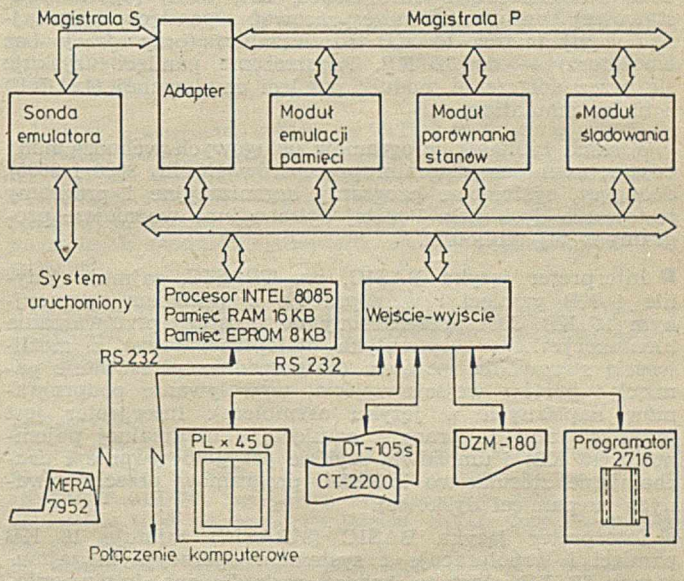
Wymienione elementy, wspomagające tworzenie oprogramowania, uzupełnione są o emulator układowy.

Koncepcja emulacji układowej polega na symulowaniu działania mikroprocesora w rzeczywistym systemie (tzn. w systemie uruchamianym) przez układ zawierający analogiczny mikroprocesor i elementy logiczne umożliwiające jego sterowanie i nadzorowanie. W procesie uruchamiania mikroprocesor zastępuje się sondą układu emulatora. Dzięki temu możliwe jest uruchamianie systemu w docelowej konfiguracji i w rzeczywistych warunkach pracy, bez konieczności wprowadzania do jego struktury specjalnych środków potrzebnych jedynie w procesie uruchamiania.

Do systemów wspomagania o takich możliwościach należy system RTDS-8 (Real Time Development System 8) opracowany w Zakładzie Systemów Automatyki Kompleksowej PAN w Gliwicach. Jest on modularnym systemem przeznaczonym do projektowania i uruchamiania zestawów, wyposażonym w uniwersalny emulator układowy mikroprocesorów 8-bitowych. Dostępny obecnie zestaw modułów i oprogramowanie systemu RTDS-8 umożliwiają projektowanie i uruchamianie zestawów opartych na mikroprocesorach INTEL 8080 i 8085.

KONFIGURACJA SPRZĘTU

System RTDS-8 składa się z dwóch podstawowych bloków funkcjonalnych: mikrokomputera i uniwersalnego emulatora mikroprocesorów 8-bitowych (rys.).



Schemat blokowy systemu RTDS-8

Blok mikrokomputera stanowi dwumodułowy (moduł procesora i moduł sprzęgający), uniwersalny system mikrokomputerowy wyposażony w procesor INTEL 8085 i zawierający co najmniej 16 KB pamięci typu RAM oraz (co najwyżej) 8 KB pamięci EPROM. Moduł sprzęgający umożliwia współpracę systemu z urządzeniami zewnętrznymi, jak np.:

- monitor ekranowy (np. MERA-7952) jako konsola systemowa
- stacja pamięci na dyskach elastycznych (PLx45D)

Mgr inż. KRZYSZTOF PLUSCZOK ukończył w 1980 r. studia na Wydziale Automatyki i Informatyki Politechniki Śląskiej. Od grudnia 1979 r. pracuje w Zakładzie Systemów Automatyki Kompleksowej PAN w Gliwicach, gdzie zajmuje się głównie oprogramowaniem emulacyjnym dla systemów wspomagających uruchomianie systemów mikroprocesorowych.

- drukarka mozaikowa (DZM-180)
- czytnik i dziurkarka taśmy papierowej (CT-2000 i DT-105s lub stacja SPTP-3)
- programator pamięci EPROM typu 2716
- system komputerowy z oprogramowaniem skrośnym — przez sprzęg szeregowy.

Integralną częścią systemu RTDS-8 jest uniwersalny emulator układowy mikroprocesorów 8-bitowych. Koncepcję emulatora oparto na oryginalnej metodzie wymuszonego wykonywania rozkazów, opisanej w pracy [3]. Możliwość wymuszonego wykonywania rozkazów jest podstawą realizacji oddziaływania na stan zasobów, sterowania stanem mikroprocesora i nadzorowania stanu systemu uruchamianego.

Poszczególne moduły emulatora stanowią układy peryferyjne o określonej strukturze portów wejścia-wyjścia i pamięci danych. Modułowa konstrukcja i programowe sterowanie przebiegiem emulacji umożliwiają traktowanie emulatora jako bloku funkcjonalnego, który może być stosowany w różnych konfiguracjach systemów wspomagania — poczynając od systemów z oprogramowaniem skrośnym, aż do autonomicznych stanowisk emulacji.

OPROGRAMOWANIE SYSTEMU RTDS-8

Oprogramowanie podstawowe zawiera monitor systemowy oraz dyskowy system operacyjny z biblioteką programów usługowych.

Monitor systemowy umieszczony w pamięci typu ROM ma możliwości typowe dla tego typu programu (jak np. monitor systemu INTELLEC MDS [2]), rozszerzone o polecenia: testowania pamięci operacyjnej, ładowania systemu operacyjnego, formatowania dyskietek, programowania pamięci EPROM (typu 2716), rozszerzenia pamięci operacyjnej o pamięć emulowaną.

Dyskowy (jednoprogramowy) system operacyjny jest zgodny (ang. *compatible*) z systemem CP/M (firmy DIGITAL RESEARCH, Pacific Grove, CA, USA) [7]. W podstawowej konfiguracji może pracować z pamięcią operacyjną o pojemności 16 KB rozszerzalnej (przy pracy bez emulatora) — do 32 KB. Rozszerzenie pamięci uzyskuje się przez dołączenie modułu pamięci emulowanej (4x4 KB) z bloku emulatora.

W skład biblioteki programów usługowych wchodzą m.in.: edytor, asembler (dla mikroprocesorów INTEL 8080 i 8085), *debugger*, systemowe programy organizacyjne i programy testowania dyskietek. Poniżej opisano inne ważniejsze programy z tej biblioteki.

- Interpreter języka BASIC, pn. TBASIC, zajmuje jedynie 4 KB pamięci i współpracuje z systemem operacyjnym 16 KB. Jest przeznaczony do prostego przetwarzania nienumerycznego (arytmetyka stałoprzecinkowa) z możliwością sterowania sprzętem (bezpośrednie adresowanie pamięci i portów wejścia-wyjścia, wywoływanie podprogramów napisanych w języku asemblera). Interpreter jest przystosowany do pracy z pamięcią o minimalnej pojemności (16 KB) i umożliwia szybkie przygotowywanie i uruchamianie stosunkowo krótkich programów przechowywanych w pamięci dyskowej.
- Interpreter języka BASIC (MBASIC) zajmuje 18 KB pamięci i współpracuje z systemem operacyjnym zajmującym 28 KB. Jest to bardzo rozbudowana implementacja języka, charakteryzująca się m.in. czterema typami zmiennych (całkowite, łańcuchowe, zmiennoprzecinkowe pojedynczej i podwójnej precyzji), dynamiczną deklaracją rozmiarów tablic, różnorodnymi funkcjami redakcyjnymi, możliwością współpracy z pamięcią dyskową i systemem operacyjnym.
- Interpreter języka FORTH [1, 6] zajmuje ok. 8 KB pamięci, ale ze względu na rozmiary obszarów roboczych, wymaga systemu operacyjnego, zajmującego 24 KB. Język, a dokładniej — system FORTH, jest uniwersalnym narzędziem do przygotowywania oprogramowania systemów mikroprocesorowych. Jego cechą charakterystyczną jest możliwość definiowania słownika najbardziej odpowiedniego dla danego zastosowania. Programy napisane w tym języku są zwięzłe i efektywne czasowo.
- Program sterujący emulacją (EM80 lub EM85, zależnie od przeznaczenia do emulacji mikroprocesorów INTEL 8080 lub 8085) zajmuje 9,5 KB pamięci operacyjnej i współpracuje z systemem CP/M zajmującym 16 KB. Jego podstawową funkcją jest koordynacja pracy wszystkich modułów funkcjonalnych emulatora. Umożliwia on operowanie zasobami systemu uruchamianego i zasobami emulatora, korzysta z funkcji systemu realizowanych sprzętowo (np. śladowanie [1]) w czasie rzeczywistym, realizacja punktów wstrzymania) oraz — zastępuje systemowi uruchamianemu oprogramowanie rezydentne, jak program ładujący lub *debugger*. Służy ponadto do organizacji procesu uruchamiania przez dialog z użytkownikiem.
- Program ładujący (ALOD) jest przeznaczony głównie do współpracy z programatorem. Służy on do przepisania pliku dyskowego (tzw. obrazu pamięci utworzonego np. za pomocą programu emulacji i zawierającego uruchamiany program) do pamięci operacyjnej. Początkowy adres ładowania jest standardowy lub podawany przez użytkownika.

UNIWERSALNY EMULATOR MIKROPROCESORÓW 8-BITOWYCH

Ogólną strukturę emulatora systemu RTDS-8 przedstawiono na rysunku 1, a zasady jego konstrukcji omówiono w pracy [4]. Podstawowymi modułami emulatora są: sonda emulowanego mikroprocesora i adapter.

Sonda jest układem o konstrukcji zależnej od typu mikroprocesora systemu uruchomianego i zawiera sam mikroprocesor wraz z logiką realizującą jego wymuszone sterowanie. Moduł wysyła sygnały mikroprocesora, bezpośrednio do podstawki mikroprocesora w systemie uruchamianym, oraz sygnały magistrali S, której organizacja umożliwia realizację elementarnych zadań procesu uruchamiania.

Adapter realizuje przejścia z magistrali S na buforowaną magistralę systemu wspomagającego (magistralę P), umożliwiającą współpracę z pozostałymi modułami emulatora, jak — moduły emulacji pamięci, porównania stanów i śladowania.

Moduł emulacji pamięci zawiera 16 KB pamięci RAM — 4 bloki po 4 KB. Bloki można dołączać do systemu uruchamianego po zadeklarowaniu adresu początkowego i typu dostępu (tylko odczyt, odczyt-zapis). Emulowaną pamięć można umieścić w obszarze adresowym już zajętym przez pamięć systemu uruchomianego. Wszystkie operacje będą wykonywane tylko na pamięci emulowanej. Zasada ta umożliwia szybkie poprawianie programu umieszczonego już w pamięci typu ROM systemu uruchamianego. Przepisuje się program z pamięci ROM do pamięci emulowanej, wprowadza poprawki i przystępuje do testowania. Z punktu widzenia systemu uruchamianego w obszarze adresowym pamięci ROM będzie występowała pamięć emulowana.

W systemie RTDS-8 można również wykorzystać pamięć emulowaną do rozszerzenia pamięci operacyjnej. Rozszerzenie pamięci i powrót do konfiguracji podstawowej odbywa się przez wydanie odpowiednich poleceń systemowych.

Moduł porównania stanów zawiera dwa analogiczne układy rozpoznawające zadany stan magistrali adresowej w określonym cyklu maszynowym. Umożliwiają one realizację wstrzymania programu wykonywanego w czasie rzeczywistym, w trybie emulacji, oraz warunkowe określenie trybu śladowania przebiegu programu.

Moduł śladowania przebiegu programu w czasie rzeczywistym umożliwia zapamiętanie 1024 cykli maszynowych. Informacje dotyczące pojedynczego cyklu obejmują: adres (16 bitów), dane (8 bitów) rodzaj cyklu (4 bity) oraz stan czterech wybranych linii dołączonych do sondy. Deklaracja warunków i początku śladowania umożliwia selektywny wybór cykli.

---

[1] Przez ślad o wanie (ang. *tracing*) rozumie się rejestrowanie śladu wykonywania programu, tj. stanu procesora i stanu magistrali w poszczególnych cyklach maszynowych.

CHARAKTERYSTYKA FUNKCJONALNA PROGRAMU EMULACJI

Oprogramowanie systemu RTDS-8 umożliwia przeprowadzenie pełnego cyklu konstrukcji oprogramowania dla systemu użytkowego. Edytor i asembler zapewniają przygotowanie tekstu programu, przetłumaczenie go i uzyskanie programu wynikowego w postaci pliku zapisanego w pamięci dyskowej.

Program emulacji (EMULATOR) służy do interakcyjnego kontynuowania procesu uruchamiania programu w konfiguracji użytkowej. Polecenia EMULATORA można podzielić na trzy podstawowe grupy funkcjonalne:

— umożliwiające przygotowanie odpowiedniej konfiguracji pamięci i załadowanie programu

— umożliwiające obserwowanie i modyfikowanie stanu zasobów systemu uruchomianego oraz kontrolowanie przebiegu programu

— umożliwiające składowanie zawartości pamięci (ang. *dumping*).

Pierwsza grupa poleceń umożliwia dołączenie bloków pamięci emulowanej przez podanie adresu i trybu dostępu do bloku. Po wykonaniu tych operacji można załadować program do pamięci systemu uruchamianego. Odpowiednie polecenia służą do ładowania programu z pamięci dyskowej (pliki typu COM, tj. bezpośrednio ładowalne, i HEX, w formacie szesnastkowym systemu INTELLEC), z komputera nadrzędnego lub z czytnika taśmy papierowej. Po załadowaniu programu przystępuje się do właściwego uruchomienia.

Za pomocą poleceń drugiej grupy można wykonywać następujące operacje:

— wyświetlanie i modyfikację stanu mikroprocesora, pamięci i portów wejścia-wyjścia

— krokowe wykonywanie programu i wyświetlanie kodów rozkazów oraz pełnego stanu mikroprocesora

— deklarowanie warunków dla modułu komparatora

— wykonywanie programu w czasie rzeczywistym od zadanego adresu (ze wstrzymaniem po spełnieniu zadanego warunku)

— sterowanie śladowaniem przebiegu programu w czasie rzeczywistym i wyświetlanie śladu

— pomiar czasu wykonywania programu.

Postać deklaracji dla każdego z dwóch komparatorów (*C1* i *C2*), rozpoznających stan magistrali adresowej w zadanym cyklu maszynowym, jest następująca:

$$C\begin{Bmatrix}1\\2\end{Bmatrix}\left[\left[\begin{Bmatrix}=\\<\\>\end{Bmatrix}<\text{adres}>\right]\left[\begin{Bmatrix}\text{OF} & \text{— } \textit{opcode fetch}\ ^{2})\\ \text{MR} & \text{— } \textit{memory read}\\ \text{MW} & \text{— } \textit{memory write}\\ \text{IR} & \text{— } \textit{input read}\\ \text{OW} & \text{— } \textit{output write}\\ \text{IA} & \text{— } \textit{instruction acknowledge}\end{Bmatrix}\right]\right]$$

Pole warunku adresowego lub warunku określającego rodzaj cyklu można pominąć. Warunków *C1* i *C2* używa się w deklaracji punktu wstrzymania programu wykonywanego w czasie rzeczywistym lub w deklaracji warunków śladowania.

Punkt wstrzymania programu może być określony przez następujące warunki:

C1 — wstrzymanie po spełnieniu zadeklarowanego warunku
C2 — jak wyżej
C+ — suma logiczna warunków *C1* i *C2*
C& — iloczyn logiczny warunków *C1* i *C2*
CP — wstrzymanie na rozkazach skoków zmieniających stan licznika rozkazów
SP — wstrzymanie na rozkazach zmieniających stan stosu
TF — zapełnienie pamięci śladu
USER — wstrzymanie na żądanie użytkownika.

Interesujące jest użycie warunków C+ i C& przy relacjach adresowych typu {<,>}. Dla $C1 < 100$, $C2 > 200$ — warunek C+ powoduje wstrzymanie programu przy dostępie poza obszar o adresach (*100, 200*). Dla $C1 > 100$, $C2 < 200$ warunek C& powoduje wstrzymanie programu przy dostępie do obszaru o adresach (*100, 200*).

Warunkowe śladowanie wymaga zadeklarowania początkowego i właściwego warunku śladowania. Po spełnieniu warunku początkowego zapamiętywane są cykle rozkazów, w których został spełniony właściwy warunek śladowania. Śladowanie zostaje wstrzymane po spełnieniu warunku wstrzymania programu. Warunki śladowania deklaruje się spośród opisanych powyżej (UNC oznacza śladowanie bezwarunkowe). W przypadku zadeklarowania warunku *CP*, jako właściwego warunku śladowania, oprócz cykli rozkazów skoku zapamiętywane są cykle następnego rozkazu.

Zawartość pamięci śladu może być wyświetlana od początku lub od dowolnego cyklu. Wyświetlany jest mnemoniczny kod rozkazu z zaznaczeniem cykli przesłań danych oraz stan wybranych linii w poszczególnych cyklach maszynowych.

Celem ostatniej grupy poleceń jest rejestracja zawartości pamięci systemu uruchamianego na zewnętrznych nośnikach informacji. Odpowiednie polecenia umożliwiają składowanie zawartości pamięci na taśmie papierowej lub w pamięci dyskowej (w postaci pliku typu COM).

* * *

W systemie RTDS-8 adapter jest przystosowany do mikroprocesorów INTEL 8080, 8085, 8048. Emulacja dowolnego mikroprocesora z tej rodziny wymaga jedynie dołączenia odpowiedniej sondy i zainicjowania odpowiedniego programu sterującego. Dotychczas opracowano sondy i programy sterujące emulacją dla mikroprocesorów INTEL 8080 i 8085.

Dalsze prace nad systemem RTDS-8 będą prowadzone w dwóch zasadniczych kierunkach. Planuje się dołączenie do systemu kompilatorów języków wysokiego poziomu — FORTRAN i PL/M — oraz rozszerzenie grupy mikroprocesorów emulowanych w systemie przez opracowanie odpowiednich modułów (adaptera, sondy) i programów sterujących.

Obecnie rozpoczyna się produkcję systemu RTDS-8 w Zakładach Urządzeń Komputerowych MERA-ELZAB w Zabrzu.

[2]) Ang. *opcode fetch* — pobranie kodu operacji, *memory read-write* — odczyt lub zapis pamięci, *input read* — odczyt wejścia, *output write* — zapis wyjścia, *instruction acknowledge* — potwierdzenie rozkazu

LITERATURA

[1] Fig-FORTH for 8080, Assembly Source Listing, Release 1.1 with Compiler Security and Variable Length Names. FORTH INTEREST GROUP, San Carlos, CA, September 1979

[2] Intellec MDS 800 Operator's Manual. INTEL Corp., Santa Clara, CA, 1975

[3] Konsek M.: Metoda instrukcji forsowanej w koncepcji stanowisk wspomagających uruchamianie systemów mikrokomputerowych. Podstawy Sterowania, 9, nr 3, str. 283—294, 1979

[4] Konsek M.: System wspomagający uruchamianie oparty na metodzie forsowania instrukcji na zasadzie blokady dostępu do pamięci. Opracowanie wewnętrzne, Zakład Systemów Automatyki Kompleksowej PAN, Gliwice, 1980

[5] Konsek M., Paulus Cz., Trojniar W.: Rozwój środków wspomagających projektowanie i uruchamianie systemów mikroprocesorowych. Opracowanie wewnętrzne, Zakład Systemów Automatyki Kompleksowej PAN, Gliwice, 1980

[6] Ragsdale W. F.: Fig-FORTH Installation Manual, Glossary Model Editor, Release 1 with Compiler Security and Variable Length Names. FORTH INTEREST GROUP, San Carlos, CA, November 1980

[7] Sobczyk M.: Dyskowy system operacyjny CP/M. INFORMATYKA, nr 2/83 (w druku).

TADEUSZ SINKIEWICZ
Instytut Maszyn Matematycznych
Warszawa

Mikroprocesorowy system wspomagania projektowania

Mikroprocesorowy System Wspomagania Projektowania (MSWP) jest uniwersalnym narzędziem przeznaczonym do projektowania, uruchamiania i testowania systemów mikroprocesorowych. Stanowi system modularny zawierający bogaty zestaw sprzętowych i programowych środków służących do projektowania i uruchamiania urządzeń mikroprocesorowych zbudowanych z układów serii INTEL 8080 i INTEL 3000 oraz ich odpowiedników (np. układów serii K580 i K589 produkcji ZSRR). Modularna budowa systemu umożliwia dostosowanie go do projektowania urządzeń opartych na mikroprocesorach innych typów. **Pod względem funkcjonalnym MSWP stanowi odpowiednik takich systemów jak INTELLEC MDS 800 firmy INTEL lub TEKTRONIX 8002 MICROPROCESOR LAB.**

KONFIGURACJA SPRZĘTU

Strukturę blokową MSWP przedstawiono na rysunku 1. Podstawowe bloki funkcjonalne znajdują się w jednostce centralnej, do której dołącza się urządzenia wejścia-wyjścia, pamięć masową na dyskach elastycznych i uruchamiany system użytkowy. Na płycie czołowej umieszczone są przyciski przerwań i wymienne gniazdo dla programowalnych pamięci stałych i układów FPLA (ang. *Field Programmable Logic Arrays).*

W zależności od potrzeb i posiadanych środków można kompletować MSWP w różnych konfiguracjach. Jest to możliwe dzięki modularnej budowie bloków funkcjonalnych oraz takiemu zaprojektowaniu układów wejścia-wyjścia, aby zapewnić możliwość stosowania różnych urządzeń zewnętrznych.

W najprostszym zestawie znajdują się następujące moduły funkcjonalne: procesor systemowy, układy sterujące urządzeniami wejścia-wyjścia, pamięć RAM o pojemności 16 KB, pamięć ROM o pojemności 2 KB, układy płyty czołowej i programator pamięci EPROM typu INTEL 1702A. Jako uniwersalne urządzenie wejścia-wyjścia można zastosować dalekopis. W ten sposób uzyskuje się prosty system do przygotowywania i uruchamiania programów dla mikroprocesorowej serii MCS 80 firmy INTEL i jej odpowiedników.

MSWP w pełnej konfiguracji jest wyposażony w pamięć RAM o pojemności 60 KB, pamięć ROM o pojemności 4 KB, pamięć masową na dyskach elastycznych (2 mechanizmy PLx45D), symulator pamięci stałych o pojemności 512 słów 64-bitowych, wkładki programatora pamięci EPROM typu 1702A, 2704, 2708, 2716, 2732 firmy INTEL, pamięci PROM typu TM 621 firmy TUNGSRAM, programator układów FPLA typu 82S100 firmy SIGNETICS, emulator układowy dla mikroprocesorów typu INTEL 8080.

Uwzględniając zapotrzebowanie użytkowników opracowano dodatkowe urządzenia (moduły) MSWP: uniwersalny moduł testujący UMT-1, monitor alfanumeryczny, oparty na standardowym odbiorniku telewizyjnym oraz lampę kasującą zapis w pamięciach EPROM za pomocą promieni ultrafioletowych. Uniwersalny moduł testujący UMT-1 służy do kontroli poprawności wykonania płytek z obwodem drukowanym i testowania pakietów z układami cyfrowymi o poziomach sygnałów wejściowo-wyjściowych zgodnych z poziomami TTL. Tester może obsługiwać do 320 styków, a standardy mechaniczne testowanych płytek i pakietów mogą być definiowane przez zamawiającego. Monitor telewizyjny umożliwia zastąpienie monitora alfanumerycznego znacznie tańszym zestawem spełniającym funkcje konsoli operatorskiej, złożonym z jednostki sterującej, klawiatury i standardowego odbiornika telewizyjnego.

Jako standardową konsolę operatorską systemu przewidziano monitor alfanumeryczny MERA 7953 z klawiaturą MERA 7946. Można zastosować również inny monitor alfanumeryczny wyposażony w sprzęg typu V24 lub drukarkę z klawiaturą DZM 180 KSR. Do pracy z taśmami dziurkowanymi przewidziano czytniki typu CT 2100 i CT 1000 oraz perforatory typu DT 105 i DT 105S.

Dodatkowym urządzeniem zewnętrznym może być drukarka DZM 180 (ze sprzęgiem równoległym), szczególnie przydatna do drukowania tekstów programów.

OPROGRAMOWANIE

Oprogramowanie systemu MSWP umożliwia tworzenie, uruchamianie i testowanie programów użytkowych przeznaczonych dla systemów mikroprocesorowych złożonych z układów serii INTEL 8080, a w wersji rozszerzonej także — serii INTEL 3000 i ich odpowiedników.

Dla zestawów MSWP bez pamięci masowej na dyskach elastycznych, oprogramowanie dostarczane na taśmie dziurkowanej umożliwia tłumaczenie, redagowanie i uruchamianie programów dla mikroprocesorów typu INTEL 8080, sterowanie pracą programatorów i symulatora pamięci stałych.

System MSWP z pamięcią na dyskach elastycznych ma znacznie bogatsze oprogramowanie podstawowe, które działa pod nadzorem dyskowego systemu operacyjnego OS-I (odpowiednik systemu operacyjnego CP/M opracowanego przez firmę DIGITAL RESEARCH, Pacific Grove, CA, USA) lub — OS-II (odpowiednik systemu ISIS II opracowanego w firmie INTEL).

W pamięci stałej jednostki centralnej rezyduje program MONITOR inicjujący i nadzorujący pracę systemu. Zawiera on również procedury obsługi urządzeń wejścia-wyjścia oraz dyrektywy ułatwiające uruchamianie programów użytkowych.

Oprogramowanie podstawowe na taśmach dziurkowanych składa się z następujących programów:

Dr inż. TADEUSZ SINKIEWICZ po ukończeniu w 1961 r. Wydziału Automatyki i Techniki Obliczeniowej Moskiewskiego Instytutu Energetycznego rozpoczął pracę w Instytucie Maszyn Matematycznych. Tematyka prac konstrukcyjnych prowadzonych w Instytucie dotyczyła konstrukcji cyfrowych układów podstawowych, urządzeń testujących i bloków maszyn cyfrowych. W roku 1974 obronił na Politechnice Warszawskiej pracę doktorską dotyczącą zagadnień modelowania dynamiki sieci przełączających. Obecnie zajmuje się zagadnieniami projektowania i zastosowań systemów mikroprocesorowych.

EDYTOR 1 — program operujący znakami, obsługujący 13 dyrektyw umożliwiających wprowadzanie, wyprowadzanie i poprawianie tekstu

EDYTOR 2 — program służący do poprawiania taśm dziurkowanych

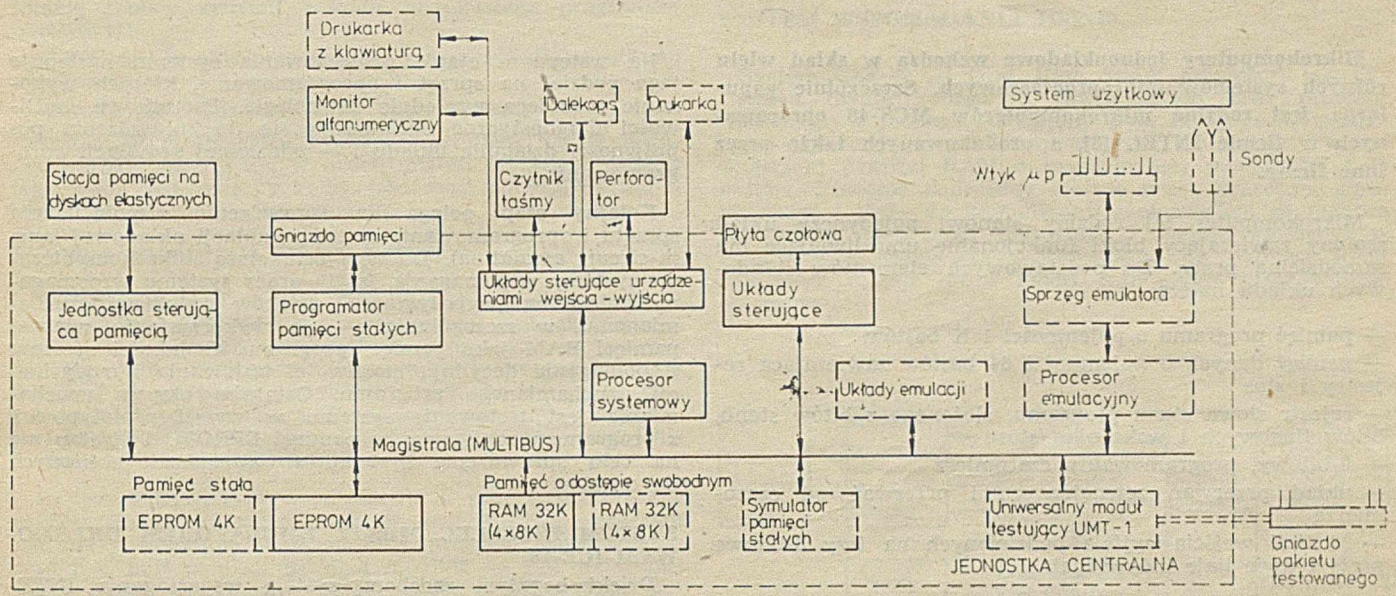
MAKROASEMBLER REM — dwuprzebiegowy program tłumaczący programy napisane w języku asemblera na kod szesnastkowy odpowiadający rozkazom mikroprocesora INTEL 8080

REMAK 80 — połączenie programu EDYTOR 1 i makroasemblera REM rezydujących równocześnie w pamięci operacyjnej i korzystających ze wspólnego bufora, co umożliwia modyfikowanie i tłumaczenie programów źródłowych bez korzystania z taśm papierowych

DYSKOWE SYSTEMY OPERACYJNE

Dyskowy system operacyjny OS-I jest zgodny z szeroko rozpowszechnionym systemem CP/M dla mikroprocesora INTEL 8080. Umożliwia on m.in. tworzenie, modyfikowanie i kasowanie plików systemowych oraz tworzenie, modyfikowanie i uruchamianie programów użytkowych. W skład podstawowego zestawu wchodzi program redagujący, asembler, program obsługi urządzeń, programy do uruchamiania, podawania stanu dyskietek, operowania plikami, wyświetlania zawartości plików i obsługi dyrektyw systemowych. Użytkownicy mogą korzystać z następujących programów dodatkowych:

— makroasembler (odpowiednik MAC)
— symboliczny *debugger* (odpowiednik SID)
— formater tekstów (odpowiednik TEX)



Struktura blokowa Mikroprocesorowego Systemu Wspomagania Projektowania

INTERPRETER języka BASIC (MINI BASIC) — realizujący operacje arytmetyczne na liczbach zmiennoprzecinkowych (stałoprzecinkowych)

EMULATOR 8080 — program służący do uruchamiania systemów opartych na mikroprocesorze INTEL 8080; dyrektywy programu umożliwiają m.in. deklarowanie bloków pamięci i urządzeń użytkowych oraz zasobów systemu, start i pracę krokową programu, operowanie zawartością pamięci i rejestrów, ustalanie trybu pracy pamięci śledzenia [1]), ustawianie punktów wstrzymania w programie

SYMROM — program obsługi symulatora pamięci stałych dla systemów z szybkimi układami mikroprocesorowymi, np. serii INTEL 3000

PROGRAMER — zespół programów nadzorujących pracę programatorów pamięci stałych.

[1]) Pamięć śledzenia umożliwia rejestrację kolejnych stanów magistrali z możliwością pełnego odtworzenia tej informacji na wybranym urządzeniu

— test dyskietki — TEST FD
— program odzyskiwania uszkodzonych plików — ODZYSK
— program sprawdzania taśmowej kopii pliku — SPRAWDŹ.

Dyskowy system operacyjny OS-II jest zgodny z systemem operacyjnym ISIS-II stosowanym w urządzeniach INTELLEC MDS firmy INTEL. Jego możliwości są analogiczne do możliwości systemu OS-I z wyjątkiem funkcji programu uruchamiania i formatera tekstów. Cenną właściwością systemu OS-II, niedostępną w systemie OS-I, jest możliwość programowania modularnego, łączenia i relokowania modułów programowych. Pod nadzorem systemu operacyjnego OS-II mogą pracować programy tłumaczące: makroasembler relokowalny (odpowiednik ASM-80) oraz kompilator języka PL/M-80.

Przedstawione moduły sprzętowe i programowe zbadano w uruchomionych egzemplarzach serii informacyjnej MSWP. Zebrane doświadczenia eksploatacyjne zostaną wykorzystane w wykonaniu kolejnej serii systemów MSWP, planowanym na pierwsze półrocze 1983 roku.

ADAM PLUTA
Przemysłowy Instytut Elektroniki NPCP
ZBIGNIEW POZNAŃSKI, JERZY SZYNKA
Instytut Technologii Elektronowej NPCP
Warszawa

Wybrane systemy wspomagania dla mikrokomputerów rodziny MCS-48

Mikrokomputery jednoukładowe wchodzą w skład wielu różnych systemów mikroprocesorowych. Szczególnie popularna jest rodzina mikrokomputerów MCS-48 opracowanych w firmie INTEL [3], a produkowanych także przez inne firmy.

Mikrokomputer tej rodziny stanowi pojedynczy układ scalony zawierający bloki funkcjonalne umożliwiające mu samodzielną pracę. Do podstawowych elementów składowych układu należą:

— pamięć programu o pojemności 1 K bajtów
— pamięć danych o pojemności 64 bajtów zawierająca rejestry i stos
— rejestr słowa stanu programu, złożonego z bitów stanu, bitów flagowych i wskaźnika stosu
— 8-bitowy, programowalny czasomierz
— układ przerwań zewnętrznych i przerwań od czasomierza
— 26 linii wejścia-wyjścia podzielonych na trzy 8-bitowe porty i dwie linie indywidualne.

Istnieje szereg odmian tego mikrokomputera różniących się pojemnością i rodzajem pamięci oraz możliwościami dołączania układów zewnętrznych. Możliwa jest także rozbudowa mikrokomputera o dodatkowe układy pamięci RAM lub ROM i układy zewnętrzne. Pamięć programu można rozszerzyć do 4 K bajtów, a pamięć danych — do 320 bajtów. Dzięki częściowej zgodności z systemem MCS-80 dopuszcza się wzajemną wymienność układów zewnętrznych.

REALIZACJA SYSTEMÓW OPARTYCH NA MIKROKOMPUTERACH JEDNOUKŁADOWYCH

Mikrokomputery jednoukładowe są przeznaczone w zasadzie do tworzenia jednomodułowych systemów mikroprocesorowych. W realizacji takich systemów występują pewne różnice w stosunku do realizacji systemów wielomodułowych. Są one konsekwencją faktu, że mikrokomputer jednoukładowy tworzy zamkniętą całość, uniemożliwiającą testowanie jej poszczególnych elementów. Wynika stąd pewna odmienność metodyki i realizacji systemów opartych na mikrokomputerach jednoukładowych.

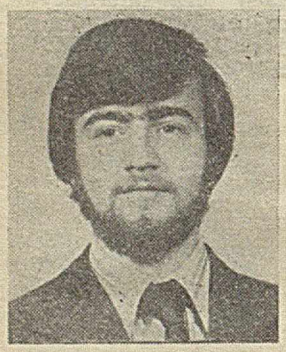
Na wstępnym etapie projektowania systemu następuje jego podział na sprzęt i oprogramowanie, których wykonanie przebiega względnie równolegle. Początkowe możliwości diagnostyczne ograniczają się do sprawdzenia poprawności działania układu i — oddzielnej symulacji oprogramowania.

Kolejny etap polega na sprawdzeniu współdziałania sprzętu i oprogramowania metodą emulacji układowej (ang. *in-circuit emulation*). Do tego celu służą mikrokomputerowe systemy wspomagania. Istota pracy systemu wspomagania polega na „wypożyczaniu" zasobów systemowi uruchamianemu, w szczególności — na wykorzystaniu własnej pamięci RAM jako pamięci programu. O jakości systemu wspomagania decydują możliwości testowania i modyfikacji uruchamianego programu. Ostatnim etapem uruchamiania jest testowanie systemu w wersji prototypowej, z programem zapisanym w pamięci EPROM. Ten etap ma na celu sprawdzenie systemu w warunkach zbliżonych do docelowych.

SYSTEM INTELLEC MDS Z EMULATOREM UKŁADOWYM ICE-48

Dzięki bogatym zasobom, system wspomagania INTELLEC MDS pracujący pod nadzorem dyskowego systemu operacyjnego, wyposażony w emulator układowy ICE 48 stwarza największe możliwości uruchamiania systemów opartych na mikrokomputerach rodziny MCS-48 [2]. Zastąpienie układu mikrokomputera w projektowanym systemie przez emulator umożliwia pracę w czasie rzeczywistym, z jednoczesnym zapamiętaniem stanu wszystkich wyjść w ciągu ostatnich 255 kroków, oraz pracę krokową lub wielokrokową z zapamiętaniem stanu wszystkich rejestrów wewnętrznych. Dzięki bogatemu oprogramowaniu systemu MDS możliwe jest wykonywanie oraz modyfikowanie programu w języku symbolicznym.

Moduł ICE-48 zawiera dwa bloki funkcjonalne: blok emulatora i blok sterujący. Blok emulatora jest wyposażony w mikroprocesor INTEL 8748 z odpowiednimi układami logicznymi do emulacji mikrokomputerów rodziny MCS-48. W jego skład wchodzi także generator impulsów o częstotliwości 6 MHz lub 3 MHz, choć można również wprowadzić zewnętrzny sygnał zegarowy. Blok sterujący zapewnia komunikację emulatora z systemem MDS przez kanał bezpośredniego dostępu do pamięci. Zawiera układy



Mgr inż. ADAM PLUTA jest absolwentem Wydziału Elektroniki Politechniki Warszawskiej (1979). Od ukończenia studiów pracuje w Przemysłowym Instytucie Elektroniki, w którym jest kierownikiem pracowni systemów mikroprocesorowych.

Mgr inż. ZBIGNIEW POZNAŃSKI jest absolwentem Wydziału MEiL Politechniki Warszawskiej, specjalność mechanika stosowana (1974). W latach 1974—1978 pracował w Instytucie Maszyn Matematycznych, zajmując się zagadnieniami symulacji cyfrowej. Obecnie jest specjalistą w Instytucie Technologii Elektronowej i zajmuje się problemami projektowania i programowania systemów mikroprocesorowych.

ustawiania punktów wstrzymania oraz generowania przerwań, a także 4 K bajty pamięci RAM, w której umieszcza się emulowany program oraz 256 bajtów zewnętrznej pamięci danych dla mikroprocesora INTEL 8748. Blok sterujący pracuje pod kontrolą mikroprocesora INTEL 8080 i umożliwia zapamiętanie stanu wyjść procesora emulacyjnego oraz odczytanie zawartości jego rejestrów, przerzutników flagowych i pamięci.

Oprogramowanie modułu ICE-48 umożliwia:

— wykonywanie programu użytkowego z możliwością ustawienia punktu wstrzymania lub krokowe wykonywanie programu z wyświetlaniem stanu wybranych rejestrów i komórek pamięci
— symulowanie przerwań systemowych
— odczytywanie i modyfikację zawartości wybranych rejestrów, portów, przerzutników, układów wyjściowych i komórek pamięci w dowolnym formacie
— wprowadzanie, wyprowadzanie, zapamiętywanie i modyfikację tablicy symboli i kodu wynikowego programów użytkowych
— definiowanie zbioru urządzeń dostępnych dla programu użytkowego
— powrót pod nadzór systemu operacyjnego.

System MDS INTELLEC wraz z modułem ICE-48 stwarza szerokie możliwości w uruchamianiu zestawów, jednak jego wadą jest znaczny koszt, jaki musi ponieść użytkownik przy zakupie takiego sprzętu.

SYSTEM PROMPT-48 FIRMY INTEL

System PROMPT-48 jest przeznaczony do programowania pamięci EPROM jednoukładowego mikrokomputera INTEL 8748 oraz do uruchamiania programów napisanych dla tego mikrokomputera. Wprowadzanie programu może odbywać się za pomocą klawiatury, kanału transmisji szeregowej lub programatora. Przez kanał transmisji szeregowej można dołączyć system PROMPT-48 do większego mikrokomputera, np. INTELLEC MDS.

System PROMPT-48 współpracuje z programatorem i z procesorem wykonującym program uruchamiany i program nadzorczy (monitor). Uruchamiany program może zajmować 1 K bajt pamięci RAM, a monitor — 4 K bajty pamięci ROM. Dodatkowo system zawiera 256 bajtów pamięci RAM stanowiącej rozszerzenie pamięci danych mikrokomputera.

Przekazanie sterowania z programu użytkowego do monitora następuje wskutek przerwania z panela sterującego. Oprócz przycisku przerwania i przycisku RESET panel jest wyposażony w klawiaturę umożliwiającą wybór rozkazu, zadawania parametrów rozkazów oraz wybór dodatkowych funkcji monitora. Kod rozkazu, parametry i wynik operacji są wyświetlane na wskaźniku cyfrowym. Do systemu można dodatkowo dołączyć zewnętrzny terminal.

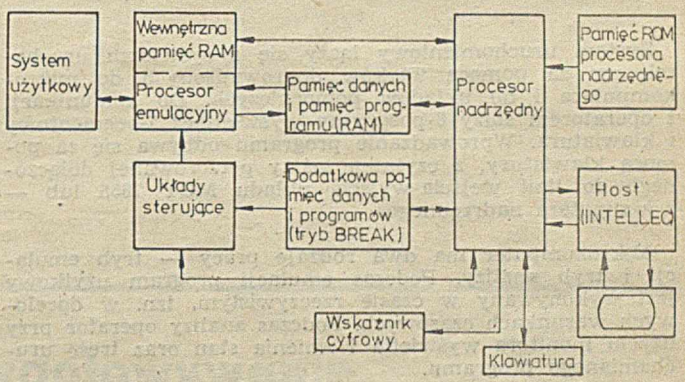
System nie ma własnego mikroprocesora, lecz istnieje możliwość współpracy z systemem INTELLEC MDS. Procesor wykonujący program użytkowy służy jednocześnie do sprawdzania jego poprawności oraz do wykonywania programu nadzorczego. Sprawdzony, poprawnie działający program można przesłać do drugiego procesora, zainstalowanego w gniazdku programatora.

Poszczególne funkcje monitora zapewniają stosunkowo łatwe wykrywanie błędów i poprawianie programu użytkowego. Umożliwiają one m.in.:

— wykonywanie programu w czasie rzeczywistym
— wykonywanie programu z wstrzymaniem (po wykonaniu każdej instrukcji bada się zawartość licznika rozkazów)
— krokowe wykonywanie programu (wykonanie kolejnej instrukcji wymaga interwencji operatora)
— sprawdzanie i modyfikowanie zawartości rejestrów, pamięci oraz portów mikrokomputera
— przesuwanie programów i danych w pamięci
— ustawianie punktów wstrzymania
— wprowadzanie i wyprowadzanie informacji kanałem transmisji szeregowej
— przepisywanie zawartości pamięci RAM do pamięci EPROM i odwrotnie oraz porównywanie zawartości odpowiednich obszarów tych pamięci
— dodawanie i odejmowanie liczb w kodzie szesnastkowym.

SYSTEM WSPOMAGANIA HSE-49

System HSE-49 [1] (ang. *High Speed Emulator*) jest systemem dwuprocesorowym zawierającym tzw. procesor emulacyjny i procesor nadrzędny (rys. 1). Procesor emulacyjny wykonuje uruchamiany program umieszczony w zewnętrznej pamięci RAM o maksymalnej pojemności 4 K bajtów. Procesor nadrzędny steruje pracą procesora emulacyjnego i służy do komunikacji z otoczeniem za pomocą klawiatury, wskaźnika cyfrowego i kanału szeregowego. Programy użytkowe są wprowadzane do pamięci RAM emulatora ręcznie lub z innego komputera przez wejście szeregowe.



Rys. 1. Schemat blokowy systemu HSE-49

Pod względem funkcjonalnym system HSE-49 stanowi rozszerzenie systemu PROMPT-48. W większości funkcje realizowane przez oba systemy są identyczne. Program nadzorczy wykonywany przez procesor nadrzędny realizuje podstawowe funkcje wymienione w opisie systemu PROMPT-48 (z wyjątkiem funkcji związanych z programowaniem pamięci EPROM), a ponadto:

— wykonywanie programu z punktami wstrzymania określonymi przez zawartość zewnętrznej pamięci RAM
— wykonywanie programu z wyświetlaniem zawartości licznika rozkazów oraz akumulatora po każdym cyklu rozkazowym (tryb AUTO-STEP) lub w każdym punkcie wstrzymania (tryb AUTO-BREAK) bez zatrzymania programu
— wykonywanie programu od adresu zerowego
— wyprowadzanie zawartości zadanego obszaru pamięci na urządzenie zewnętrzne (terminal)
— wprowadzanie i wyprowadzanie informacji w specjalnym formacie (tzw. *INTEL HEX FILE FORMAT*) przez kanał transmisji szeregowej.

Istotna jest możliwość zmiany, konfiguracji systemu HSE-49 za pomocą zwor lub innych dodatkowych elementów. W szczególności można rozszerzać pamięci RAM dla programu i danych — do 4 K bajtów każdą.

Dr inż. JERZY SZYNKA jest absolwentem Wydziału Elektroniki Uniwersytetu Technicznego w Dreźnie. Pracę podjął w Instytucie Maszyn Matematycznych, gdzie zajmował się komputerowymi systemami sterowania. Od 1978 r. pracuje w Instytucie Technologii Elektronowej w zespole oprogramowania systemów mikroprocesorowych.

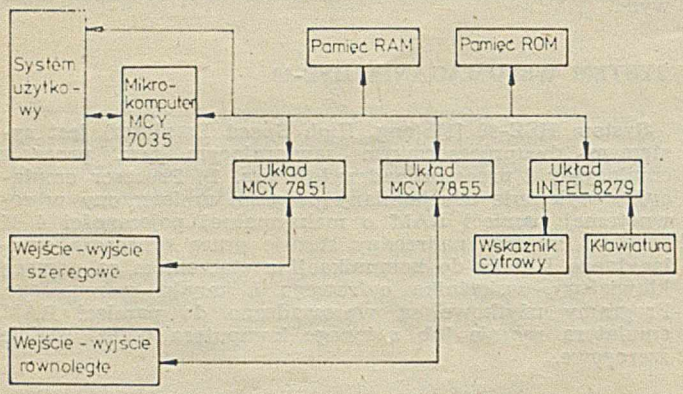
SYSTEM URUCHOMIENIOWY EMU-48 DLA MIKROKOMPUTERÓW RODZINY MCY 7848/35

System służący do uruchamiania i testowania programów oraz emulacji układowej w czasie rzeczywistym dla mikrokomputerów rodziny MCY 7848/35 (rys. 2) został opracowany w Instytucie Technologii Elektronowej NPCP [4]. Jest to system jednoprocesorowy zawierający poza mikrokomputerem MCY 7035 (odpowiednik INTEL 8035) i jego blokami funkcjonalnymi:

— 2 K bajty pamięci RAM przeznaczonej na uruchamiany program

— 256 bajtów pamięci RAM przeznaczonej na dane

— układy pomocnicze MCY 7855, MCY 7851 (odpowiednio: INTEL 8255 i 8251) i INTEL 8279.

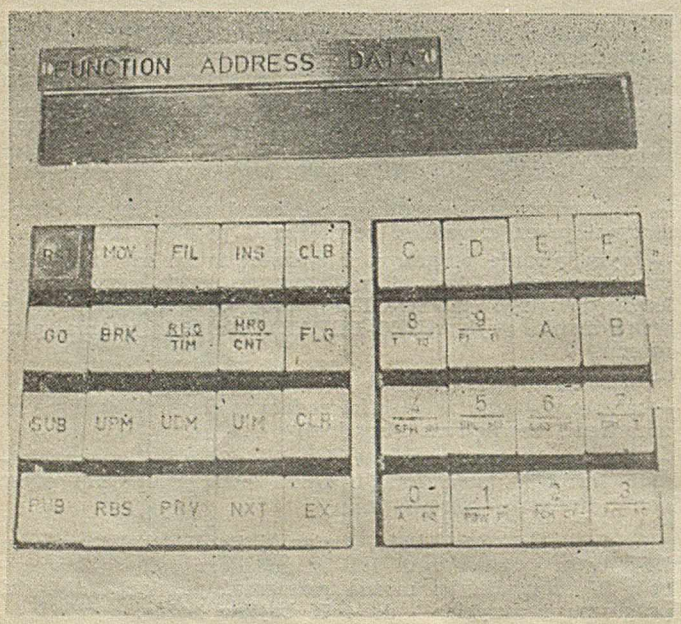


Rys. 2. Schemat blokowy systemu uruchomieniowego EMU-48 dla mikrokomputerów rodziny MCY 7848/35

System uruchomieniowy łączy się z systemem uruchamianym za pomocą wtyków doprowadzonych do mikrokomputera i do układów pomocniczych. Do komunikacji z operatorem służy 8-pozycyjny wyświetlacz 7-segmentowy i klawiatura. Wprowadzanie programu odbywa się za pomocą klawiatury, z czytnika taśmy perforowanej dołączonego do linii wejścia-wyjścia układu MCY 7855 lub — z komputera nadrzędnego.

Mikrokomputer ma dwa rodzaje pracy — tryb emulacji i tryb analizy. Podczas emulacji program użytkowy jest wykonywany w czasie rzeczywistym, tzn. w docelowych warunkach czasowych. Podczas analizy operator przy użyciu monitora wyświetla i zmienia stan oraz treść uruchamianego programu.

Fragment pulpitu sterującego systemu, zawierającego klawisze podstawowych funkcji



Ze względu na ograniczone możliwości sprzętowe tego emulatora narzucono szczególnie duże wymagania na jego oprogramowanie. Monitor (system operacyjny) zajmujący 2 K bajty pamięci EPROM, oprócz większości funkcji programowych opisanych systemów, spełnia niektóre funkcje realizowane w nich sprzętowo (np. obsługa punktów wstrzymania).

Do podstawowych funkcji systemu zalicza się (fot.):

— przeglądanie i zmianę zawartości komórek pamięci użytkowych (pamięci programu, wewnętrznej i zewnętrznej pamięci danych — polecenia SUB, UPM, UDM, UIM), zawartości rejestrów programowych R0-R7 z obu zestawów rejestrów (polecenie REG), zawartości rejestrów sprzętowych (polecenie HRG), stanu przerzutników flagowych (FLG), oraz punktów wstrzymania (BRK)

— wykonywanie programu w czasie rzeczywistym z możliwością podania adresu początkowego i parametrów czasomierza (polecenia TIM/CNT)

— wykonywanie programu z punktami wstrzymania i przechowywaniem zawartości wszystkich rejestrów, wewnętrznej i zewnętrznej pamięci RAM, stanu przerzutników flagowych itp. (polecenia GO, BRK)

— przesuwanie zawartości pamięci (programu lub danych) wewnętrznej lub zewnętrznej (polecenie MOV oraz UPM, UDM, UIM)

— wypełnianie stałą obszaru pamięci programu lub danych (FIL)

— wprowadzanie programu użytkowego z taśmy perforowanej do pamięci (polecenie INS)

— usuwanie wszystkich lub wybranych punktów wstrzymania (polecenie CLB).

Porównanie możliwości funkcjonalnych systemów wspomagania

Funkcje	PROMPT—48	HSE—49	Emulator MCY 7848/35
Programowanie pamięci EPROM	+	–	–
Modyfikowanie i przeglądanie rejestrów i pamięci programu	+	+	+
Modyfikacja i przeglądanie wewnętrznej pamięci RAM	+	+	+
Modyfikacja i przeglądanie zewnętrznej pamięci RAM i przerzutników flagowych	–	–	+
Ustawianie i kasowanie punktów wstrzymania	+	+	+
Wykonywanie programu w czasie rzeczywistym	+	+	+
Wykonywanie programu z punktami wstrzymania	+	+	+
Wykonywanie programu z wyborem trybu pracy czasomierza	–	–	+
Krokowe wykonywanie programu	+	+	+
Przesuwanie danych i programów w pamięci	+	–	+
Wypełnianie zadanego obszaru pamięci stałą	–	+	+
Wprowadzanie i wyprowadzanie zawartości rejestrów, danych lub programów kanałem transmisji szeregowej			
Wprowadzanie danych kanałem transmisji równoległej	+	+	–
Wyszukiwanie określonego bajtu w zadanym obszarze pamięci	–	–	+
Obliczanie sumy i różnicy liczb w kodzie szesnastkowym	+	–	–
Wykonywanie programu z punktami wstrzymania określonymi przez dostęp do zewnętrznej pamięci RAM	+	+	–
Wyprowadzanie zawartości zadanego obszaru pamięci na terminal	–	+	–
Zmiana zestawu rejestrów przy przeglądaniu	–	+	–
Przerywanie wykonywania programu w dowolnej chwili	–	–	+
Emulacja układowa	+	+	–
	–	+	+

Monitor spełnia ponadto szereg funkcji pomocniczych, jak:

— zerowanie wyświetlaczy z jednoczesnym zgłoszeniem się systemu (polecenie CLR)
— odtwarzanie poprzedniej wartości danej lub adresu, podczas wprowadzania nowej (RUB)
— automatyczna zmiana zestawu rejestrów (RBS)
— przeglądanie zawartości rejestrów i pamięci w obu kierunkach (PRV, NXT).

Niezależnie od funkcji programowych emulator ma kilka funkcji realizowanych sprzętowo. Należą do nich:

— inicjowanie działania (polecenie RST)
— krokowe wykonywanie programu
— ustawianie niektórych przerzutników flagowych
— przełączanie kierunku przepływu danych przez port P2 w zależności od sposobu jego wykorzystania. Istnieje ponadto możliwość realizacji transmisji szeregowej.

* * *

Porównanie możliwości funkcjonalnych omówionych systemów wspomagania przedstawiono w tabeli. Nie uwzględniono w niej emulatora ICE-48, który w połączeniu z systemem typu MDS stanowi urządzenie przewyższające pod każdym względem wszystkie pozostałe. Ze względu na znaczny koszt takiego urządzenia jego zakup jest uzasadniony tylko w przypadku intensywnego wdrażania systemów mikroprocesorowych. Należy bowiem wziąć pod uwagę fakt, że koszt systemu wspomagania wpływa w istotny sposób na koszt realizacji całego projektu systemu mikroprocesorowego.

W większości zastosowań przydatne są urządzenia typu PROMPT-48, HSE-49 lub system uruchomieniowy dla mikrokomputerów rodziny MCY 7848/35. Wskutek kompromisu między ograniczeniami wynikającymi z prostej konstrukcji a pożądanymi możliwościami funkcjonalnymi systemu, uzyskano skuteczne i tanie rozwiązanie, przydatne dla szerokiego kręgu użytkowników. Stosunkowo największe możliwości ma system HSE-49, który jest najbardziej złożony pod względem sprzętowym. Zaletą systemu PROMPT-48 jest właściwość programowania pamięci EPROM, natomiast wadą znacznie ograniczone możliwości emulacji.

System wspomagania dla mikrokomputerów MCY 7848/35 stanowi najprostsze rozwiązanie konstrukcyjne przy zapewnieniu większości podstawowych funkcji do uruchamiania i testowania programów oraz emulacji układowej. Jego ograniczenia, będące naturalną konsekwencją prostej budowy, są porównywalne z ograniczeniami systemu HSE-49. Zasadniczą zaletą tego systemu w stosunku do produktów firmy INTEL będzie jego dostępność na rynku krajowym.

LITERATURA

[1] A High-Speed Emulator for Intel MCS-48 Microcomputers. Application Note AP-55A, INTEL Corp., Santa Clara, CA
[2] Jankowski M. T.: Systemy wspomagania MDS — narzędzie uruchamiania i badań systemów mikrokomputerowych. Systemy mikroprocesorowe, s. 4, str. 5, 1980
[3] MCS-48 Family of Singlechip Microcomputers — User's Manual. INTEL Corp., Santa Clara, CA, July 1978
[4] Pluta A., Poznański Z., Szynka J.: System uruchomieniowy dla mikrokomputerów rodziny MCY 7848/35. Systemy mikroprocesorowe, 1982 (w druku).

TADEUSZ GÓRNICKI, MAGDALENA SZEŻYŃSKA
Instytut Podstaw Elektroniki
Politechnika Warszawska

Środki wspomagające przygotowywanie i uruchamianie oprogramowania mikrokomputerów jednoukładowych MCS-48

Mikrokomputery jednoukładowe są przeznaczone do specjalizowanych zastosowań, wymagających spełnienia prostych i ograniczonych ilościowo funkcji kontrolnych. Charakteryzują się niewielką pojemnością pamięci programu i pamięci danych oraz wymagają oszczędnego korzystania z linii (portów) wejścia-wyjścia. Umożliwiają dołączanie dodatkowych, zewnętrznych układów pamięci (kosztem pewnego ograniczenia dostępnych linii wejścia-wyjścia) oraz rozbudowę portów. Nie wystarcza to jednak do tworzenia uniwersalnego systemu uruchomieniowego w oparciu o mikroprocesor jednoukładowy. Powstaje zatem konieczność posługiwania się oprogramowaniem skrośnym — tak w fazie przygotowania (pisania) programu użytkowego, jak i jego uruchamiania [2].

W fazie przygotowywania programów, oprogramowanie wspomagające powinno ułatwiać pisanie i poprawianie tekstu źródłowego, musi być więc wygodne i proste w obsłudze. Konieczna, w takim przypadku, współpraca systemu z kilkoma urządzeniami zewnętrznymi (monitor ekranowy, drukarka, czytnik, perforator) powoduje wykorzystanie części linii wejścia-wyjścia, a programy redagujące i tłumaczące zajmują zwykle duże obszary pamięci.

W fazie uruchamiania urządzenia mikroprocesorowego zawierającego mikrokomputer jednoukładowy nie można zebrać pełnej informacji o stanie procesora, jeżeli program jest wykonywany w czasie rzeczywistym. Przesyłanie informacji dokonywane wewnątrz układu nie może być zarejestrowane przez urządzenia zewnętrzne. Przy krokowym wykonywaniu programu możliwe jest odczytanie stanu wewnętrznego procesora, ale wymaga to zastosowania specjalizowanych układów zewnętrznych o dość rozbudowanej logice.

Uwzględniając powyższe ograniczenia, można utworzyć minimalny zestaw środków wspomagających programowanie i uruchamianie urządzeń zawierających mikrokomputery jednoukładowe. Zestaw ten powinien zawierać:

- niezbędny, w fazie przygotowywania programu użytkowego, edytor programów symbolicznych i ich translator (asembler)
- narzędzie uruchamiania, tj. — w zależności od potrzeb — symulator, prosty system mikroprocesorowy (zawierający mikrokomputer) lub emulator tego mikrokomputera w innym systemie uruchomieniowym
- urządzenie wpisujące ostatecznie przygotowany program do pamięci programu mikrokomputera (programator).

W artykule, na tle ogólnej charakterystyki mikroprocesorowych systemów uruchomieniowych, przedstawiono konkretną realizację opracowaną w Instytucie Podstaw Elektroniki Politechniki Warszawskiej. W skład systemu, zgodnego ze standardami firmy INTEL, wchodzą: edytor, asembler, symulator i programator mikrokomputera jednoukładowego INTEL 8048 [1]. Zrealizowano go na uniwersalnym zestawie opartym na procesorze INTEL 8080, o minimalnej pojemności pamięci 8 K bajtów.

PRZYGOTOWYWANIE PROGRAMÓW

Faza przygotowywania programów dla mikrokomputerów jednoukładowych jest identyczna jak dla mikroprocesorów ze stałą listą rozkazów. Składa się z dwóch etapów: redagowania programu źródłowego oraz jego tłumaczenia na kod maszynowy.

Redagowanie programu

Pierwszy etap jest realizowany przez program redagujący (edytor), który umożliwia zapis (na nośniku informacji, np. na taśmie papierowej, taśmie magnetycznej, dysku elastycznym, w buforze pamięci) oraz poprawianie tekstu programu wprowadzanego przez operatora z klawiatury. W przypadku mikrokomputerów jednoukładowych wykorzystuje się istniejące programy redagujące większych, uniwersalnych systemów wspomagających projektowanie urządzeń mikroprocesorowych. Większość z nich działa w trybie konwersacyjnym i umożliwia wymianę części tekstu z dokładnością do pojedynczych znaków.

W Instytucie Podstaw Elektroniki opracowano tzw. edytor ekranowy. Tekst programu jest wpisany do bufora, a ekran stanowi ruchome „okno" do przeglądania jego zawartości. Znaki sterujące ruchem znacznika (ang. *cursor*) w górę, w dół, w prawo i w lewo umożliwiają naprowadzenie „okna" na odpowiedni fragment programu oraz znacznika — na dowolny znak w tekście. Polecenia usunięcia linii lub znaku czy rozsunięcia linii lub znaków, w połączeniu z możliwością dopisania dowolnego tekstu w miejscu oznaczonym znacznikiem, umożliwiają łatwe i skuteczne poprawianie programów. Edytor napisany w języku asemblera mikroprocesora INTEL 8080, zajmuje około 1,5 K bajtów pamięci.

Tłumaczenie programu

Drugim etapem jest tłumaczenie programu symbolicznego na kod maszynowy. Ze względu na ograniczoną pojemność pamięci mikrokomputerów jednoukładowych język asemblera jest w zasadzie jedynym językiem symbolicznym, w którym można je programować.

Program tłumaczący (asembler), poza automatycznym tłumaczeniem programów, powinien zapewniać wykrywanie błędów, a także dostarczać informacji niezbędnych do dalszego uruchamiania programów. W przypadku mikrokomputerów jednoukładowych stosuje się asemblery skrośne, zrealizowane najczęściej na uniwersalnych systemach wspomagających. Są one zazwyczaj zgodne (ang. *compatible*) pod względem formatów tekstu źródłowego, kodu wynikowego, wydruku programu, sygnalizacji błędów itp. z całym oprogramowaniem podstawowym określonej firmy.

Asembler skrośny, wchodzący w skład omawianego systemu uruchomieniowego, umożliwia:

— symboliczne zapisywanie rozkazów wewnętrznych procesora

— adresowanie symboliczne i definiowanie symboli w programie (dyrektywy EQU, SET)

— statyczną organizację pamięci, definiowanie bajtów i słów pamięci (dyrektywy ORG, END, DS, DB, DW).

Stwarza on możliwość stosowania różnych zapisów liczb, wyrażeń arytmetycznych, stałych tekstowych jako argumentów itp.

W czasie tłumaczenia programu źródłowego asembler tworzy:

— kod wynikowy programu na określonym nośniku informacji

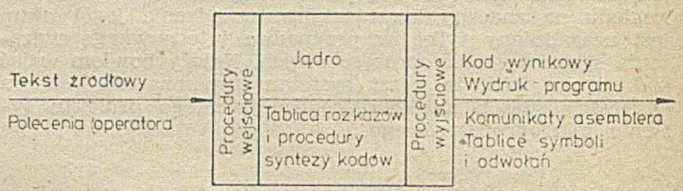
— wydruk programu z mapą pamięci i sygnalizacją błędów

— tablicę symboli

— tablicę odwołań.

Mapa pamięci ułatwia śledzenie wykonywania programu w trakcie jego uruchamiania oraz modyfikację programu w pamięci maszyny. Tablice symboli i odwołań zwiększają czytelność wydruku i ułatwiają wyszukiwanie fragmentów programu.

Asembler dla mikrokomputera INTEL 8048 zajmuje około 4,5 K bajtów pamięci, a ponadto wymaga niewielkiego obszaru pamięci RAM dla dynamicznie tworzonych tablic symboli i odwołań. Najważniejszą cechą omawianej realizacji jest łatwość adaptacji do innych typów mikrokomputerów jednoukładowych (innych języków), dzięki modułowej strukturze asemblera (rys. 1).



Rys. 1. Budowa asemblera

Jądro jest częścią stałą, niezależną od listy rozkazów mikrokomputera i zawiera podstawowe procedury asemblera, jak np. obliczanie wartości argumentów, definiowanie symboli, przeszukiwanie tablic itp. Od typu procesora zależy tylko część zmienna, w skład której wchodzi tablica odpowiadająca liście rozkazów procesora oraz podprogramy syntezy kodu, modyfikujące kod podstawowy rozkazu na podstawie wartości argumentów. Wyodrębnienie procedur wejścia-wyjścia ułatwia przystosowanie asemblera do innej konfiguracji sprzętu.

URUCHAMIANIE PROGRAMÓW

W fazie uruchamiania podstawowymi urządzeniami wspomagającymi są symulatory, proste (np. jednopłytkowe) systemy uruchomieniowe oraz emulatory. Możliwości narzędzia uruchomieniowego, niezależnie od jego rodzaju, są związane z architekturą mikrokomputera i jego funkcjami. W tabeli 1 przedstawiono proponowany zestaw poleceń symulatora mikrokomputera jednoukładowego rodziny MCS-48, odpowiadający (z pewnymi uzupełnieniami) także innym urządzeniom uruchomieniowym.

Symulator

Symulator realizujący powyższe polecenia, napisany w języku asemblera, zajmuje 8 KB pamięci systemu INTEL 8080. W obszarze tym mieści się 4 KB pamięci programu, 256 bajtów zewnętrznej pamięci danych i wszystkie rejestry wewnętrzne mikrokomputera, symulowane w pamięci systemu. Przykładową sekwencję poleceń przedstawiono na rys. 2.

Zestaw poleceń symulatora umożliwia załadowanie do pamięci programu przechowywanego na taśmie papierowej, rozpoczęcie wykonywania programu od dowolnego adresu i wykonywanie go aż do określonego punktu wstrzymania. Umożliwia on także — w trybie pracy krokowej — przeglądanie i zmianę wskazanego bajtu pamięci programu, pamięci danych lub rejestrów oraz wyperforowanie zawartości pamięci programu na taśmie. Na taśmie papierowej można również przechowywać zawartości pamięci danych i rejestrów. W związku ze specyficznym sposobem adresowania pamięci przez mikrokomputer za pomocą rejestru, symulator umożliwia dostęp do pamięci danych i re-

Tabela 1. Polecenia symulatora mikrokomputerów MCS-48

A (adres)	usunięcie punktu wstrzymania
B (adres)	ustawienie punktu wstrzymania
DDM (adr 1), (adr 2) DEDM (adr 1), (adr 2) DPM (adr 1), (adr 2)	wyświetlenie zawartości pamięci danych, zewnętrznej pamięci danych lub pamięci programu od adresu 1 do adresu 2
F (adres)	perforacja adresu końcowego z możliwością podania adresu startowego programu po załadowaniu
G (adres)	symulacja od podanego adresu
K	kontynuacja symulacji
LDM LEDM LPM	załadowanie pamięci danych, zewnętrznej pamięci danych i pamięci programu
M	koniec symulacji
NB	usunięcie wszystkich punktów wstrzymania
PDM	wyprowadzenie na perforator zawartości pamięci danych
PEDM (adr 1), (adr 2) PPM (adr 1) (adr 2),	wyprowadzenie na perforator zawartości zewnętrznej pamięci danych lub pamięci programu od adresu 1 do adresu 2
QI (port) = (adres) QO (port) = (adres)	zadeklarowanie adresu procedury obsługi portu
RESET	rozpoczęcie symulacji od wyzerowania mikrokomputera
SDM (adres) SEDM (adres) SPM (adres)	modyfikacja zawartości bajtu pamięci danych, zewnętrznej pamięci danych lub pamięci programu
SSOFF SSON	wyłączenie i włączenie trybu krokowego
X (rejestr)	modyfikacja zawartości rejestru
X*a* (rejestr)	modyfikacja zawartości pamięci danych wskazywanej przez rejestr
XX*a* (rejestr)	modyfikacja zawartości zewnętrznej pamięci danych wskazywanej przez rejestr

```
*DPMC,4
   0000   27 00 27  04  02
*SPM
   0000   27-23 00 - 27 - 17  04 -
*B2
*RESET
 A    R0   R1  CR0  CR1      PSW  F0 F1 RB MB  PC  IR
00    40   60  40   9B   10001000   0  0  0  0  002  23   000002
01    40   60  40   9B   10001000   0  0  0  0  002  04   000003
02    40   60  40   9B   10001000   0  0  0  0  002  04   000003M
*XA
02-35
*K
35    40   60  40   9B   10001000   0  0  0  0  002  04   000008
36    40   60  40   9B   10001000   0  0  0  0  002  04   000011R
   R2   R3   R4   R5   R6   R7
   44   4C   44   44   40   44   M
*NB
*SSON
*K
 A    R0   R1  CR0  CR1      PSW  F0 F1 RB MB  PC  IR
36    40   60  40   9B   10001000   0  0  0  0  002  04   000011
37    40   60  40   9B   10001000   0  0  0  0  003  17   000012
37    40   60  40   9B   10001000   0  0  0  0  002  04   000014
38    40   60  40   9B   10001000   0  0  0  0  003  17   000015M
*SSOFF
*PPMC,4
*T
*
```

Rys. 2. Sekwencja poleceń symulatora

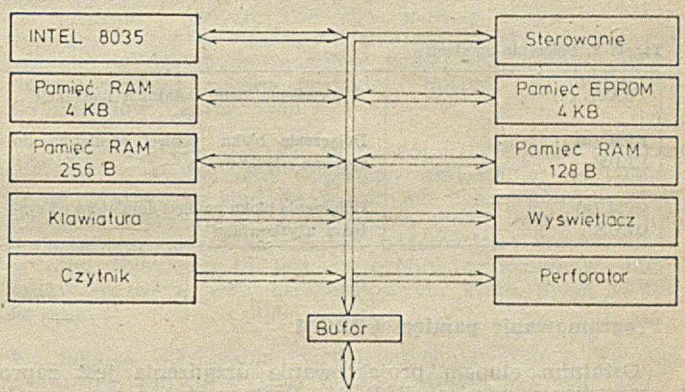
jestrów na podobnej zasadzie, tj. przez podanie rejestru zawierającego adres poszukiwanego bajtu. Polecenia te, zaczynające się od znaku X, uzupełniają zestaw typowych poleceń modyfikacji i ułatwiają przeprowadzanie symulacji przez użytkownika.

Ze względu na specyfikę zastosowań mikrokomputerów rodziny MCS-48, istotną cechą symulatora jest możliwość zobrazowania stanu linii wejścia-wyjścia i dołączenia podprogramów symulujących specjalizowane urządzenia zewnętrzne. Symulator ma własne podprogramy obsługi portów wejścia-wyjścia, wywoływane przy każdorazowej zmianie stanu portów lub przy użyciu rozkazów wejścia. W takim przypadku na konsoli operatorskiej pojawia się informacja o stanie portu w zapisie szesnastkowym, poprzedzona (w razie potrzeby) komunikatem, którego portu dotyczy. Dane wejściowe wprowadza się po odpowiednim komunikacie również w zapisie szesnastkowym. Niezależnie od tego, można dołączyć do symulatora własne podprogramy obsługi portów, zastępujące odpowiednie wywołania w symulatorze. Podprogramy te należy umieścić w obszarze nie zajmowanym przez symulator i zadeklarować za pomocą poleceń typu Q.

Symulator nie jest (i nie może być) programem całkowicie autonomicznym — ze względu na konieczność wykorzystania standardowych urządzeń wejścia-wyjścia systemu, w którym jest używany. W omawianym przypadku zaistniała ponadto konieczność dostosowania sposobu symulacji specjalizowanych wejść testowych T0 i T1 oraz przerwań do możliwości systemu użytkowego. Tym niemniej symulator napisano w sposób ułatwiający przenoszenie go z systemu na system przy minimalnych zmianach programowych.

Pozostałe narzędzia uruchamiania

Symulator, jako narzędzie w pełni programowe, nie umożliwia dynamicznego testowania uruchamianego urządzenia — w czasie rzeczywistym. Jest to natomiast możliwe w systemach wykorzystujących ten sam typ mikrokomputera, co urządzenie uruchamiane.

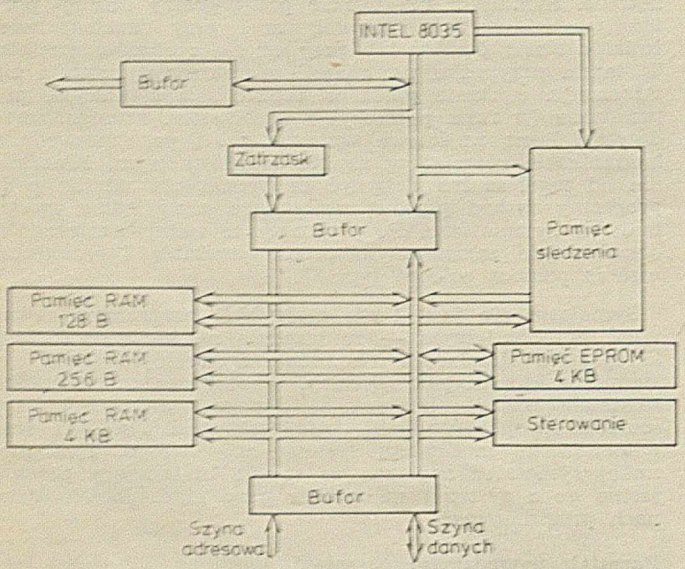


Rys. 3. Schemat blokowy prostego systemu wspomagającego uruchamianie urządzeń zawierających mikrokomputery MCS-48

Jednym z rozwiązań jest samodzielny mikrokomputer ze specjalizowaną klawiaturą i wyświetlaczem. Na rysunku 3 przedstawiono przykładowy schemat blokowy takiego komputera. Ten sam procesor (np. INTEL 8035) wykonuje program użytkowy i realizuje polecenia systemowe. Wszystkie linie wejścia-wyjścia mikrokomputera muszą być buforowane w taki sposób, aby z punktu widzenia dołączonego do nich urządzenia uruchamianego (o ile nie jest wykonywany program użytkowy) mikrokomputer znajdował się w stanie zatrzymania. Na stan linii wejścia-wyjścia nie powinny wtedy wpływać żadne przesłania dokonywane pod kontrolą programu systemowego.

Najbardziej rozwiniętym narzędziem do uruchamiania urządzeń mikroprocesorowych, także — zawierających mikrokomputery jednoukładowe, są emulatory. Umożliwiają one zastąpienie procesora urządzenia uruchamianego przez procesor emulatora, pozostający pod kontrolą systemu, oraz wypożyczenie części zasobów systemu temu urządzeniu.

Koncepcję budowy emulatora mikrokomputerów rodziny MCS-48 ilustruje schemat blokowy przedstawiony na rys. 4. Ważną częścią emulatora jest moduł śledzenia i rejestracji stanu procesora podczas pracy w czasie rzeczywistym. Na informację o stanie składa się zawartość licznika rozkazów, kod rozkazu, stan portów oraz linii T0, T1 i INT.



Rys. 4. Emulator mikrokomputerów rodziny MCS-48

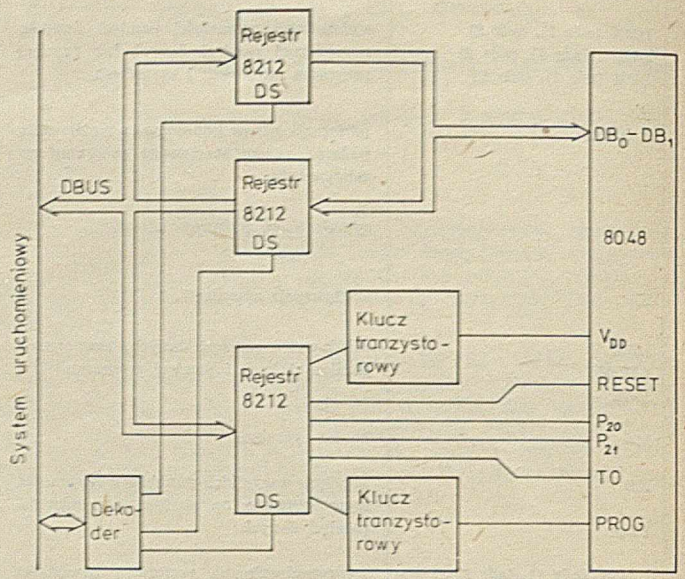
Jedno z poleceń emulatora powoduje wyświetlenie tej informacji. Odpowiednie operowanie zasobami emulatora zapewniają inne polecenia (tab. 2).

Tabela 2. Polecenie emulacji

TRACE (n)	Wyświetlenie śladu ostatnich instrukcji
EPM (n) EEDM	Dołączenie bloku pamięci emulatora do systemu użytkownika
QPM (n) QEDM	Odłączenie bloku pamięci emulatora od systemu użytkownika

Programowanie pamięci EPROM

Ostatnim etapem projektowania urządzenia jest zaprogramowanie pamięci EPROM. Programator można zrealizować w postaci autonomicznego urządzenia lub dodatkowego pakietu sterowanego bezpośrednio z systemu uruchomieniowego, jak to zrobiono w omawianym przypadku (rys. 5).



Rys. 5. Schemat blokowy programatora

Układ składa się z trzech rejestrów typu 8212, dwóch kluczy tranzystorowych, inwerterów i buforów typu 7406 i 7407 oraz przerzutnika monostabilnego typu 74123. Program sterujący służy do zaadresowania programowanej pamięci, przesłania danych do zaprogramowania i wyzwolenia przerzutnika generującego impuls programujący.

Możliwe jest zaprogramowanie pamięci EPROM zawartością zadeklarowanego obszaru pamięci, tzw. bufora. Do bufora należy uprzednio wpisać informację z taśmy papierowej lub z pamięci EPROM mikrokomputera (kopiowanie). W czasie programowania dokonuje się weryfikacji kolejno zaprogramowanych bajtów z sygnalizacją ewentualnych błędów.

* * *

Przedstawiony zespół środków jest konieczny do projektowania urządzeń mikroprocesorowych, a dla mikrokomputerów jednoukładowych praktycznie — wystarczający. Przy jego użyciu przetestowano oprogramowanie autonomicznego monitora graficznego opartego na mikroprocesorze INTEL 8048 oraz oprogramowanie specjalizowanego kalkulatora do obliczeń wyników testów psychologicznych.

LITERATURA

[1] INTEL Corp.: MCS-48 User's Manual. Santa Clara, CA, USA
[2] Kojemski A.: Oprogramowanie skrośne dla mikroprocesorów. Systemy mikroprocesorowe, nr 7, str. 5, 1981.

ZASADY PRENUMERATY

Zamówienia i przedpłaty na prenumeratę INFORMATYKI przyjmuje: Wydawnictwo SIGMA, skrytka 1004, 00-950 Warszawa; konto bankowe: nr 1036-7490-139-11 III O/M NBP Warszawa.

Instytucje przesyłają zamówienia, zawierające: liczbę zamawianych egzemplarzy, okres prenumeraty, pełny adres zamawiającego, adresy odbiorców oraz numer konta bankowego zamawiającego. Dopisek PRENUMERATA STAŁA przy zamówieniu całorocznym oznacza, że zamawiający nie musi już ponawiać zamówień, lecz jedynie dokonywać corocznie przedpłaty wg obowiązujących cen.

Prenumeratorzy indywidualni dokonują przedpłaty przekazem na ww. konto, podając na odwrocie odcinka: liczbę egzemplarzy oraz okres prenumeraty.

Terminy przedpłat: do 25 listopada — I kwartał, I półrocze, cały następny rok (w tym prenumerata stała); do 10 marca — II kwartał; do 10 czerwca — III kwartał, II półrocze; do 10 września — IV kwartał. Prenumerata kwartalna — 225 zł, półroczna — 450 zł, roczna — 900 zł. Prenumerata ze zleceniem wysyłki za granicę jest dwukrotnie droższa.

Dodatkowych informacji udziela: Dział Handlowy Wydawnictwa SIGMA — Warszawa, ul. Mazowiecka 12, tel. 26-80-16.

MAREK TADEUSZ JANKOWSKI
Instytut Technologii Elektronowej NPCP
Warszawa

KRABUS – sprzęg wewnątrzkasetowy dla systemów mikrokomputerowych

Przez sprzęg (ang. interface) rozumie się na ogół granicę między współpracującymi modułami komputerowymi, a ściślej — definiujący tę granicę zbiór zasad określających sposób połączenia, wykaz i parametry wymienianych sygnałów oraz protokoły przesyłania informacji, umożliwiające współpracę różnych jednostek lub urządzeń. W praktyce jednak nieodzowne jest jednoznaczne zdefiniowanie, poza wymienionymi zasadami, różnych właściwości konstrukcyjnych sprzęgu, a zwłaszcza:

— rodzaju stosowanych złącz

— rozmieszczenia sygnałów na złączach.

Upowszechnienie standardowych rozwiązań [1, 2, 12] następuje zarówno oficjalnie, wskutek działalności różnych organizacji międzynarodowych, jak i nieformalnie — na skutek dominacji czołowych producentów sprzętu komputerowego. Dotyczy to zarówno sprzęgów równoległych — wewnątrz kaset (por. niżej), między kasetami (np. Unibus firmy DEC, BxP Bus firmy INTEL) i między urządzeniami (np. GPIB [1]) zdefiniowana w normie IEEE-488 [2])) — jak i sprzęgów szeregowych, dopuszczających transmisję na większe odległości (np. CCITT [3]) V-24 albo EIA [4]) RS 232C i RS 422).

W przypadku systemów mikrokomputerowych szczególne znaczenie mają sprzęgi wewnątrzkasetowe, łączące pakiety umieszczane w kasecie. Wskutek znacznego zwiększenia stopnia scalenia mikroukładów, pojedyncze pakiety (ang. *board*, płytka) stanowią dość dobrze zdefiniowane, zamknięte i w pewnym sensie samodzielne bloki funkcjonalne, moduły (np. moduły nadrzędne ang. *master*, jak procesor, sterownik DMA [5]), oraz — podległe, ang. *slave*, jak pamięci RAM, EPROM, układy wejścia-wyjścia itp.), dla których konieczna jest normalizacja zasad współpracy. Ponadto sprzęgi wewnątrzkasetowe wymagają dodatkowego określenia:

— wymiarów stosowanych pakietów

— wartości napięć zasilających oraz ich rozmieszczenia na złączach — analogicznie do sygnałów.

Równocześnie, w znacznym stopniu niezależnie, normalizuje się konstrukcje mechaniczne kaset i pakietów. W Europie najważniejszy jest obecnie tzw. standard Eurokarty, znormalizowany przez IEC [6]) i przyjęty do stosowania w krajach RWPG [7, 14].

Sygnały sprzęgów wewnątrzkasetowych można podzielić na sygnały sterujące, adresy i dane. W zasadzie są one przesyłane równoległymi szynami systemowymi (zwanymi magistralą), łączącymi wszystkie moduły kasety. Wyjątkiem jest tu standard CAMAC Komitetu ESONE [7]), w którym do modułów doprowadzone są również linie indywidualne (np. LAM) od sterownika znajdującego się w kasecie. W nowszych sprzęgach często występują wspólne, przełączane linie adresów i danych. Tak więc moduły można w zasadzie umieszczać w dowolnym miejscu kasety, co nie ma wpływu na dostęp do modułów podległych, a może jedynie wpłynąć na zmianę priorytetu dostępu do magistrali dla modułów nadrzędnych.

Paradoksem jest jednak, że rozwój normalizacji stwarza potencjalną groźbę coraz większego zamieszania w tej dziedzinie. Wynika to stąd, że w ramach poszczególnych grup sprzęgów pojawiają się opracowania spełniające identyczne funkcje, lecz oparte na różnych podstawach i konkurujące ze sobą. Zjawisko to jest szczególnie widoczne w przypadku sprzęgów wewnątrzkasetowych, które mogą różnić się m.in.:

- zasadami sterowania wydzielonych modułów pamięci i układów wejścia-wyjścia (synchronizacja z pracą mikroprocesora, rozróżnianie pamięci i portów, rozróżnianie operacji zapisu i odczytu)
- możliwością lub brakiem możliwości bezpośredniego dostępu mikroprocesorów do wydzielonych modułów
- sposobem transmisji danych i adresów (szyny jedno- lub dwukierunkowe, przełączane lub nie).

Sprzęgi wewnątrzkasetowe mogą mniej lub bardziej odpowiadać architekturze określonego mikroprocesora. Tak więc wybór typu mikroprocesora i sprzęgu ma wpływ na złożoność układów pośredniczących między lokalnymi szynami mikroprocesora a magistralą kasety.

PRZEGLĄD SPRZĘGÓW WEWNĄTRZKASETOWYCH

Do najbardziej rozpowszechnionych sprzęgów wewnątrzkasetowych należą powstałe w połowie lat siedemdziesiątych, rozwiązania: S-100 (firmy MITS) oraz MULTIBUS (firmy INTEL), choć istnieją także inne opracowania pochodzące z tego okresu, jak VERSABUS firmy MOTORO-

[1]) GPIB — General Purpose Interface Bus
[2]) IEEE — Institute of Electrical and Electronics Engineers
[3]) CCITT — Commiteé Consultatif International Telegrafique et Telefonique UIT (Union International des Telecomunications)
[4]) EIA — Electronic Industries Association

[5]) DMA — Direct Memory Acces
[6]) IEC — International Electrotechnical Commission
[7]) ESONE — European Standards On Nuclear Electronics

Dr inż. MAREK TADEUSZ JANKOWSKI ukończył studia w 1957 r. na Wydziale Łączności Politechniki Warszawskiej, ze specjalnością elektronika. W roku 1962 otrzymał dyplom Imperial College w Londynie, a w 1972 r. obronił doktorat w Politechnice Warszawskiej. Pracował w Instytucie Maszyn Matematycznych, gdzie był projektantem podstawowych układów maszyn cyfrowych i systemu sterowania dla Wytwórni Polipropylenu w płockiej Petrochemii oraz w Zakładzie Budowy Maszyn Matematycznych Politechniki Warszawskiej, gdzie był projektantem maszyny do badań biomedycznych ANOPS, za co otrzymał II Nagrodę w konkursie Mistrz Techniki 1966. Od roku 1977 pracuje w Instytucie Technologii Elektronowej Naukowo-Produkcyjnego Centrum Półprzewodników, gdzie zajmuje się zastosowaniami systemów mikroprocesorowych, ze szczególnym uwzględnieniem inteligentnych terminali graficznych.

LA, STD-Bus firmy PROLOG oraz MUBUS powstały na Uniwersytecie w Lozannie. Obydwa wymienione sprzęgi opracowano dla najpopularniejszego w tym czasie mikroprocesora INTEL 8080. Miarą ich obecnego rozpowszechnienia w krajach rozwiniętych jest wielkość i zakres produkcji zgodnych ze standardem (również pod względem wymiarów pakietów i typów złącz pakietów) pamięci, modułów graficznych, analogowych itp. Istnieją także pakiety oparte na mikroprocesorach firm ZILOG, MOTOROLA, i in., zbudowane w standardzie S-100, ale również — MULTIBUS, stanowiącym opracowanie konkurencyjnej firmy.

Standard S-100 [10], którego nazwa pochodzi od stu linii magistrali, opracowano przed wprowadzeniem sterownika systemowego INTEL 8228. Z tego względu ma on zbyt duży i niezbyt dogodny zbiór sygnałów sterujących, jedynie 8-bitowe, jednokierunkowe szyny danych (wejściową i wyjściową) i tylko 16 linii adresowych. Niekorzystne jest zastosowanie napięć zasilających 8 V i 18 V, zamiast bardziej typowych dla układów TTL i MOS napięć 5 V i 12 V, co powoduje konieczność lokalnej stabilizacji napięć bezpośrednio na płytkach. Mimo tych wad, standard S-100, po zastosowaniu go w mikrokomputerze ALTAIR, bardzo rozpowszechnił się na rynku amerykańskim — w komputerach osobistych (ang. *personal computers*). Prace nad jego pełną normalizacją podjęła niedawno organizacja IEEE (projekt P696).

Standard MULTIBUS [9] opracowano przy założeniu stosowania sterownika INTEL 8228 w systemach opartych na mikroprocesorze INTEL 8080. Firma INTEL produkuje także specjalne adaptery sprzęgu — typ 8218 (dla mikroprocesora INTEL 8080) oraz typ 8219 (dla mikroprocesora INTEL 8085). Standard MULTIBUS zastosowano m.in. w firmowych systemach wspomagania MDS i PROMPT, a także w popularnej rodzinie modułów SBC (Single Board Computer), a ostatnio wprowadzono do produkcji adapter typu 8288 dla mikroprocesorów iAPX86 i 88.

Sygnały informacyjne oraz napięcia zasilające, ±5 V i ±12 V, są rozmieszczone na 86-stykowym złączu krawędziowym (na dodatkowym 60-stykowym złączu krawędziowym rozmieszczono sygnały opcjonalne). Rozszerzony MULTIBUS ma 16-bitową, dwukierunkową szynę danych i 20-bitową szynę adresową, a zarazem umożliwia pracę wieloprocesorową w kasecie. Omawiany sprzęg jest przedmiotem prac normalizacyjnych organizacji IEEE (projekt P796), a także staje się — przynajmniej w sensie elektrycznym i logicznym — standardem krajów RWPG.

Ostatnio pojawiło się wiele nowych propozycji normalizacyjnych dotyczących sprzęgów wewnątrzkasetowych opracowanych m.in. przez DIN [8]) (norma 66264), Komitet 47B IEC (systemy VME oraz AMS), IEEE (projekt P896) oraz ESONE (system E3S). Wszystkie te propozycje uwzględniają — podobnie jak MULTIBUS — możliwość tworzenia systemów wieloprocesorowych oraz stosowanie mikroprocesorów 16-bitowych. Ponadto dopuszcza się stosowanie mikroprocesorów 24-bitowych (E3S) oraz 32-bitowych (VME i P896) z odpowiednio większą liczbą linii adresowych. W odróżnieniu od MULTIBUS najnowsze propozycje charakteryzują się:

- zastosowaniem standardowych pakietów o wymiarach podwójnej Eurokarty (160 × 233,4 mm albo 220 × 233,4 mm); wyjątkiem jest norma DIN zalecająca stosowanie pojedynczej Eurokarty 160 × 100 mm
- zastosowaniem złącz pośrednich 64- lub 96-stykowych (DIN zaleca dwa złącza 32-stykowe).

Najbardziej dyskusyjna jest propozycja DIN, która nie stwarza możliwości opracowywania systemów rozbudowanych. Wprawdzie umożliwia ona stosowanie mikroprocesorów 16-bitowych, ale adresowanie jest ograniczone tylko do 16 K słów pamięci. Ponadto, zdaniem autora, pojedyncza Eurokarta stwarza zbyt duże ograniczenia konstrukcyjne.

Propozycja IEC pn. AMS [6, 13], z elektrycznego i logicznego punktu widzenia, zawiera w sobie MULTIBUS. Poza rozmieszczeniem sygnałów na dwu 96-stykowych złączach pośrednich, istotnym novum jest wprowadzenie dodatkowej szyny pn. RESIDENT BUS odpowiadającej mikroprocesorowi INTEL 8086. Druga propozycja IEC (VME) [5], którą popierają firmy konkurencyjne w stosunku do INTELA (MOSTEK, MOTOROLA i SIGNETICS/PHILIPS), jest wprawdzie nowszą i bardziej rozbudowaną niż MULTIBUS, ale w rzeczywistości — przy większej liczbie linii adresowych i danych oraz przy wszelkich różnicach w sterowaniu — raczej nie wnosi niczego istotnie nowego.

Natomiast propozycje IEEE i ESONE 3,8 wyraźnie wybiegają w przyszłość — w kierunku tworzenia zestawów wieloprocesorowych, o większym stopniu złożoności i mają szanse stać się standardami lat osiemdziesiątych. Należy sądzić, że założono w nich stosowanie najnowszych typów mikroprocesorów (np. INTEL 8086), gdyż występuje tylko jedno napięcie zasilające ±5 V oraz wspólna przełączana szyna adresowa i danych (zamiast częściej stosowanych dwu oddzielnych szyn — adresowej i danych). Są to sprzęgi bardziej złożone niż np. MULTIBUS, i w praktyce zrezygnowano w nich z szeroko jeszcze stosowanego, ale nienajnowszego mikroprocesora INTEL 8080 (który wymaga kilku napięć zasilających i ma rozdzielone szyny adresów i danych).

Zestawienie linii danych, adresowych i zasilania dla omówionych standardów oraz wybranych mikroprocesorów firmy INTEL przedstawiono w tabeli 1.

PROPOZYCJA KRAJOWEGO SPRZĘGU WEWNĄTRZKASETOWEGO KRABUS

Na tle międzynarodowej działalności w dziedzinie normalizacji sprzęgów wewnątrzkasetowych krajową sytuację charakteryzuje brak koordynacji oraz niepotrzebne powtarzanie analogicznych prac. W Polsce opracowano zaskakująco wiele rodzin pakietów mikrokomputerowych na poziomie modeli lub prototypów, z których żadna (autor miał możliwość zapoznać się z odpowiednimi pracami w co najmniej dziesięciu różnych instytucjach) nie jest zgodna (ang. *compatible*) z jakąkolwiek inną. Praktycznie żaden z systemów nie jest produkowany seryjnie, a w dodatku część z nich jest niewłaściwie ukierunkowana z konstrukcyjnego punktu widzenia.

[8]) DIN — Deutches Institut für Normung

Tabela 1. Zestawienie linii danych, adresowych i zasilania dla wybranych mikroprocesorów i sprzęgów standardowych

Nazwa	Linie danych	Linie adresowe	Zasilanie [V]	Uwagi
Mikroprocesory;				
INTEL 8080	8	16	±5, +12	
INTEL 8085	8	8+8	+5	Przełączana szyna danych
INTEL 8088	8	8+12	+5	Przełączana szyna danych
INTEL 8086	16	16+4	+5	Przełączana szyna danych
Sprzęgi standardowe;				
S-100	8	16	+8, +18	Dwie jednokierunkowe szyny danych
MULTIBUS	16	20	±5, ±12	
DIN 66264	16	16	+5, ±15	
IEC VME	32	32	+5, ±12	Dodatkowo RESIDENT BUS z przełączaną szyną danych
IEC AMS	16	23	±5, ±12	
IEEE P896	32	32	+5	Przełączana szyna adresów i danych
ESONE E3S	24	24	+5	Przełączana szyna adresów i danych
KRABUS	16 (8)	23 (16)	±±5, ±12	Dla wersji z pojedynczym złączem 64-stykowym

Przykładowo — w części krajowych opracowań stosuje się jeszcze tzw. mały pakiet riadowski, albo przestarzały standard pakietów CAMAC. Choć w niektórych opracowaniach zastosowano tzw. podwójną Eurokartę, „płytszą" (160 mm) lub „głębszą" (220 mm), występuje w nich dodatkowe zróżnicowanie z racji wprowadzenia złącz krawędziowych lub pośrednich, 64- lub 84-stykowych. Z elektrycznego i logicznego punktu widzenia najczęściej stosuje się rozmaicie zredukowany MULTIBUS (niekiedy także z niewielkimi rozszerzeniami), ale rozmieszczenie sygnałów i napięć zasilających jest, oczywiście, wzajemnie niezgodne.

Ponadto niektóre systemy nie zapewniają możliwości pracy wieloprocesorowej. W jednym przypadku wprowadzono standard MUBUS [12] mający nieco inne niż MULTIBUS sygnały sterujące, a w innym rozwiązaniu występują dwie jednokierunkowe szyny danych zamiast pojedynczej szyny dwukierunkowej.

W dziedzinie standaryzacji sprzęgów istnieje zatem w Polsce wyraźna luka, spowodowana zarówno brakiem zaawansowania odpowiednich prac normalizacyjnych, jak i brakiem producenta pakietów i kaset, który z racji poziomu produkcji, a także zaplecza systemowo-programistycznego, mógłby narzucić swój standard. W związku z tą sytuacją opracowano przedstawioną poniżej propozycję krajowego standardu wewnątrzkasetowego.

Propozycja sprzęgu KRABUS uwzględnia istniejące rozwiązania i tendencje światowe, ale także (a może przede wszystkim) realia krajowe. Stanowi wypadkową różnych i często sprzecznych uwarunkowań, niemniej autorowi wydaje się, że w dużym stopniu je godzi.

Podstawowe założenia tego sprzęgu są następujące:

- pakiet podwójnej Eurokarty (169 × 233,4 mm albo równoprawnie 220 × 233,4 mm), a zatem — standard mający już prawo obywatelstwa w całej Europie

- złącza pośrednie 64-stykowe, prostsze w porównaniu z krawędziowymi ze względu na wykonanie druków oraz tolerancję grubości laminatów, produkowane przez Zakłady ELTRA

- zestaw sygnałów i napięć zasilających — taki jak sprzęgu MULTIBUS, który jest dostosowany do mikroprocesora INTEL 8080, zarówno ten sprzęg, jak i mikroprocesor są rozwiązaniami obecnie najbardziej rozpowszechnionymi w Polsce.

Ponieważ w oryginalnym sprzęgu sygnały są rozmieszczone na złączu 86-stykowym, do najistotniejszych zagadnień należy sposób rozmieszczenia sygnałów i napięć zasilających na dwu złączach 64-stykowych. W związku z tym proponuje się następujące założenie:

- rozmieszczenie wszystkich sygnałów i napięć zasilających dotyczących mikroprocesora 8-bitowego na pojedynczym złączu 64-stykowym, natomiast umieszczenie pozostałych sygnałów (tzn. rozszerzanie magistrali dla mikroprocesorów 16-bitowych i wprowadzenie niektórych sygnałów opcjonalnych) na drugim złączu.

Dzięki temu, w przypadku mikroprocesorów 8-bitowych istnieje możliwość stosowania tylko jednego złącza, co prowadzi do oczywistych oszczędności, a zarazem umożliwia wykorzystanie identycznego standardu w przypadku pojedynczej Eurokarty.

W tabeli 2 przedstawiono propozycję rozmieszczenia sygnałów i napięć zasilających, według powyższych założeń. Poza sygnałami MULTIBUS, których oznaczenia są w zasadzie zgodne z tym standardem wprowadzono dodatkowo kilka sygnałów sprzęgu IEC AMS:

MMAP ϕ —2 — przewidziane do określania (kodowania) rodzaju pamięci dla adresów powyżej 64 K bajtów

INT-Aϕ — INT-A2 — rezerwowe linie przerwań, np. do kodowania sterowników przerwań połączonych kaskadowo

AUXϕ-3 — rezerwowe linie pomocnicze

BTMO — sygnał time-out, dotyczący braku transmisji na szynach

BPAR ϕ-1 — bity parzystości dla dwóch bajtów danych.

Złącza i elementy znajdują się po prawej stronie pakietów (patrząc w głąb kasety, tzn. od strony płaszczyzny wkładania pakietów). Przyjęto następujące typy złącz produkcji Zakładów ELTRA: wtyk typu 811 064 0935 0001, gniazdo typu 821 064 0915 0001.

Tabela 2. Propozycja sprzęgu wewnątrzkasetowego KRABUS dla systemów mikrokomputerowych

Złącze górne			Złącze dolne		
rząd a	styk	rząd c	rząd a	styk	rząd c
GND	1	GND	GND	1	GND
+5 V	2	+5 V	+5 V	2	+5 V
+5 VB[1])	3	+5 VB[1])	+5 V	3	+5 V
+12 V	4	+12 V	+12 V	4	+12 V
BCLK	5	INIT	+15 V	5	+15 V
BPRN[2])	6	BPRO[2])	+24 V	6	+24 V
BUSY	7	BREQ[2])	GND	7	GND
MRDC	8	MWTC	DAT 14	8	DAT 15
IORC	9	IOWC	DAT 12	9	DAT 13
XACK	10	INH 1	DAT 10	10	DAT 11
—	11	INH 2	DAT 8	11	DAT 9
CCLK	12	CBRQ	MMAP 2	12	—
INT 6	13	INT 7	MMAP 0	13	MMAP 1
INT 4	14	INT 5	ADR 22	14	BHEN
INT 2	15	INT 3	ADR 20	15	ADR 21
INT 0	16	INT 1	ADR 18	16	ADR 19
ADR 14	17	ADR 15	ADR 16	17	ADR 17
ADR 12	18	ADR 13	BPAR	18	BPAR 1
ADR 10	19	ADR 11	INTA	19	INTA-A2
ADR 8	20	ADR 9	—	20	—
ADR 6	21	ADR 7	INT-A0	21	INT-A1
ADR 4	22	ADR 5	AUX 2[1])	22	AUX 3[2])
ADR 2	23	ADR 3	AUX[1])	23	AUX 1[2])
ADR 0	24	ADR 1	PFIN	24	PFSN
DAT 6	25	DAT 7	BTMO[1])	25	MPRO[2])
DAT 4	26	DAT 5	RESE[1])	26	HLT[2])
DAT 2	27	DAT 3	+5 VB	27	+5 VB
DAT 0	28	DAT 1	GND	28	GND
−12 V	29	−12 V	−15 V	29	−15 V
−5 V	30	−5 V	−12 V	30	−12 V
+5 V	31	+5 V	−5 V	31	−5 V
GND	32	GND	GND	32	GND

[1]) Dodatkowe napięcie +5 V z akumulatora (spoza magistrali)
[2]) Linie indywidualne pakietów.

Należy podkreślić, że sprzęg KRABUS, przedstawiony w tabeli 2, stanowi podzbiór sprzęgu IEC AMS, i to nie tylko w sensie zestawu sygnałów, ale również ze względu na rozmieszczenie tych sygnałów na identycznych stykach odpowiednich złącz. Jedyne różnice polegają na tym, że:

- inne jest rozmieszczenie napięć zasilających na górnym złączu (aczkolwiek na tych samych polach, tzn. na stykach 1—4 oraz 29—32), dopasowane do mikroprocesora INTEL 8080 i umożliwiające stosowanie dla tego mikroprocesora tylko jednego złącza

- ze względu na bazę elementową o niezbyt dużym stopniu scalenia dopuszcza się stosowanie pakietów o wymiarach 220 × 233,4 mm obok pakietów 160 × 233,4 mm

- rzędy „b" złącz 96-stykowych są usunięte, choć wymiary proponowanych wtyków i gniazd 64-stykowych są identyczne jak 96-stykowych.

W przyszłości możliwe będzie zatem bezpośrednie przejście na pełny standard sprzęgu IEC AMS, z zastosowaniem złącz 96-stykowych. Jest to celowe ze względu na dopasowanie do rozwoju bazy elementowej (np. wprowadzenie mikroprocesora INTEL 8086), a także wykorzystanie technologii wielowarstwowych pakietów drukowanych. Na etapie stosowania mikroprocesora INTEL 8080 jest to nieuzasadnione zarówno z technicznego, jak i ekonomicznego punktu widzenia, gdyż wymagałoby stosowania dwu złącz 96-stykowych zamiast jednego 64-stykowego.

* * *

Przedstawiona propozycja powstała ze względu na obecne potrzeby i wobec braku jakiegokolwiek krajowego standardu sprzęgu wewnątrzkasetowego. Ten brak sprawił, że prace nad sprzęgami wewnątrzkasetowymi zaczęto prowadzić równolegle w szeregu ośrodków, co zresztą pośrednio wynika z zamieszczonego powyżej krótkiego przeglądu. Autor ma nadzieję, że przyjęcie proponowanego standardu w jakimś nieformalnym „klubie użytkowników" mogłoby przyspieszyć i właściwie ukierunkować krajowe prace nad normalizacją sprzęgów, gdyż dłużej już nie można czekać.

Propozycja sprzęgu KRABUS została opracowana i uzgodniona wspólnie z dr. inż. A. Syryczyńskim z Przemysłowego Instytutu Automatyki i Pomiarów; również mgr inż. J. Maryan z Politechniki Warszawskiej doszedł do identycznych w pewnym zakresie wniosków.

LITERATURA

[1] Balph T., Kister J.: UP Bus Gears up to a 32-bit Future. Electronic Design, 5 July 1980
[2] Borrill L. P.: Microprocessor Bus Structures and Standards. EUROMICRO 1980
[3] ESONE Committee: Standard Microprocessor System For Data Acquisition, Processing and Control. E3S, Luxemburg, 1981
[4] Hans Knürr AG: Mechanik für die Elektronik Eurocard 2100 (katalog)
[5] IEC Subcommittee 47B: Proposal of the German National Committee for the VME Bus Specification. IEC 47B (Germany) 3, April 1982
[6] IEC Subcommittee 47B: Proposal of the German National Committee for the AMS Bus System. IEC 47B (Germany) 5, May 1982
[7] IEC Subcommittee 48D: Dimensions of Subracks and Associated Plug-in Units for Mounting in Electronic Equipment According to IEC Publication 297. (Sec. ed.)
[8] IEEE Computer Society: P896 - A Proposed Standard Backplane Bus Specifications for Advanced Microcomputer Systems, Draft 4.1.2, 2 February 1982
[9] INTEL Corp.: Intel MULTIBUS Interfacing. Application Note AP-28A, Santa Clara, CA
[10] Lesea A., Zaks R.: Microprocessor Interfacing Techniques. SYBEX, Paris, 1978
[11] Nicoud J. D.: Mubus Standard. Micro Scope, Vol. 1, No. 8, April 1977
[12] Papenberg L. R., Rydham M.: Versatile Memory Bus Handles Mixed Memories Compatibly, Electronic Design, Vol. 27, No. 18, 1 September 1979
[13] Siemens AG: AMS — The Microcomputer System for Multicomputer Applications, Siemens Components, Vol. 16, No. 7, March 1981
[14] Standard SZW tema 17.082.38-79, Pridłożenie 5.1. Tipowyje konstrukcii, tipy i osnownyje razmiery.

Dodatkowo, warto zwrócić uwagę Czytelników na interesujący artykuł: Kapliński C., Expandable Microprocessor Bus for Distributed Processing, Electronic Components and Applications, Vol. 4, No 4, August 1982, oraz — nowe specjalistyczne czasopismo „Interfaces in Computing", wydawane przez Elvier Sequoia w Szwajcarii (przyp. aut.).

Tekst został wydrukowany bez uzgodnienia części poprawek redakcyjnych i terminologicznych z Autorem. (Red.)

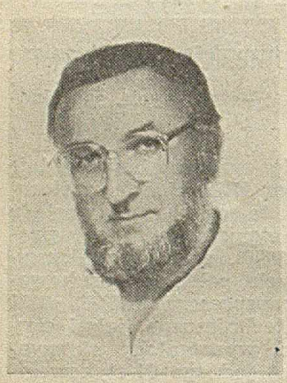
JERZY DAŃDA
Instytut Technologii Elektronowej
Naukowo-Produkcyjne Centrum Półprzewodników
Warszawa

Oprogramowanie systemów mikroprocesorowych — praktyczne aspekty polityki producenta półprzewodników

Postępy w konstrukcji i technologii układów scalonych powodują bardzo szybkie zmniejszanie kosztów sprzętu informatycznego. Procesowi temu nie towarzyszy odpowiednie zmniejszanie kosztu oprogramowania. Ocenia się, że koszt opracowania programów stanowi 60—80% łącznych kosztów opracowania systemu informatycznego. Przynajmniej z tego względu, wprowadzaniu mikroprocesorów na rynek krajowy powinny towarzyszyć wieloaspektowe działania producenta systemów mikroprocesorowych w dziedzinie oprogramowania.

Zespół zajmujący się oprogramowaniem systemów mikroprocesorowych w pionie mikroelektroniki Instytutu Technologii Elektronowej (ITE) prowadził działania mające na celu obniżenie społecznych kosztów tworzenia oprogramowania. Zdecydowano, że dla osiągnięcia tego celu należy:

- **zastosować metody programowania powszechnie uznawane za efektywne**
- **stworzyć przesłanki techniczno-organizacyjne do zminimalizowania możliwości dublowania prac programistycznych**
- **przygotować potencjalnych użytkowników systemów mikroprocesorowych do szybkiego rozpoczęcia własnych prac nad programowaniem.**



Mgr inż. JERZY DAŃDA ukończył Politechnikę Warszawską w 1958 r. Jest współkonstruktorem pamięci rtęciowej jednej z pierwszych elektronicznych maszyn cyfrowych wykonanych w Polsce w 1959 r. oraz autorem ponad siedemdziesięciu artykułów technicznych i popularyzatorskich z dziedziny elektronicznej techniki obliczeniowej. Od ponad 20 lat jest członkiem zespołu Instytutu Maszyn Matematycznych, a obecnie — szefem Zespołu Oprogramowania Systemów Mikroprocesorowych w Instytucie Technologii Elektronowej CEMI.

MODELOWANIE STRUKTURALIZOWANE

Pierwsze zadanie zrealizowano przez opracowanie metody programowania będącej pochodną podejścia zwanego programowaniem strukturalnym [10]. Metoda ta, nazwana modelowaniem strukturalizowanym [4], polega na podzia-

le procesu tworzenia programu na dwie fazy. W pierwszej fazie tworzy się model obiektu sterowanego i programów sterujących, w drugiej — właściwe programy.

W pierwszej fazie stosuje się język wysokiego poziomu (najczęściej SIMULA 67 [11], w drugiej — język symboliczny mikroprocesora. W ten sposób łączy się wygodę programowania w języku wysokiego poziomu z jednoczesnym spełnieniem specyficznych wymagań systemów mikroprocesorowych, jak zajętość pamięci i szybkość działania programu. W większości przypadków wymagania te można spełnić tylko przez napisanie ostatecznej wersji programu w języku symbolicznym.

Warto zaznaczyć, że utworzenie modelu obiektu sterowanego znacznie przyspiesza prace przy uruchamianiu programów, gdyż model ten stanowi naturalne źródło sygnałów testujących.

Omawianą metodę zweryfikowano pisząc m.in. program elektronicznej gry w bilard [1, 2], która stanowiła prosty model procesów współbieżnych, oraz program dla sterownika pieca dyfuzyjnego [15] (w ramach współpracy z Przemysłowym Instytutem Elektroniki NPCP). Uproszczony wariant metody stosowano również przy pisaniu dużo prostszych programów, służących do przetwarzania wyników testów psychologicznych, tzw. kwestionariuszy osobowości Eysencka i Cattella [5, 12].

Ze względu na trudny dostęp do komputerów mających translatory SIMULI, opracowano pakiet programów symulacyjnych [13, 14], napisanych w języku PL/M [9].

BIBLIOTEKA B.ITE.M I ZAGADNIENIA PERSPEKTYWICZNE

Drugie zadanie realizowano przez przygotowanie programów mających szeroki zakres zastosowań, np. podprogramów operacji arytmetycznych (zmienno- i stałoprzecinkowych), procedur konwersji danych, procedur podstawowych operacji tekstowych itp. **Opracowano system informatyczny do zautomatyzowania operacji adminstracyjno-edytorskich utworzonej biblioteki programów mikroprocesorowych B.ITE.M** (Biblioteka ITE — Mikroprocesorowa [6, 8]. Przygotowanie tego systemu i prowadzenie wstępnej eksploatacji biblioteki powierzono w ramach kooperacji przedsiębiorstwu ZETO w Łodzi, z którym zawarto wieloletnią umowę. W ramach tej umowy wykonano m.in. konwersacyjny system MIPL, umożliwiający redagowanie, translację skrośną i symulację programów, napisanych w języku PL/M [9]. Komputerami macierzystymi (ang. *host computer*) systemu MIPL są maszyny JS EMC, jednakże na życzenie klientów autorzy mogą go odpowiednio adaptować, instalując na dowolnym komputerze wyposażonym w translator języka FORTRAN.

Jeśli użytkownicy systemów mikroprocesorowych zaczną korzystać z biblioteki zarówno czynnie, wzbogacając jej zbiory swoimi programami, jak i biernie, tzn. sprawdzając, czy w zbiorach znajduje się potrzebny program i nabywając go (zamiast podejmować opracowanie własnego programu), to będzie można uznać, że akcja powiodła się. Sukces będzie zależeć zapewne przede wszystkim od odpowiednich rozwiązań ekonomicznych [17], a także od jakości programów, znajdujących się w bibliotece.

Programy przekazane przez ITE celem zapoczątkowania biblioteki powstały w czasie weryfikowania metody modelowania strukturalizowanego. W związku z tym wydaje się, że zostały przetestowane wystarczająco dokładnie i usunięto z nich większość błędów. Nowe programy, nabywane do zbiorów bibliotecznych będą odpowiednio testowane, bądź przyjmowane na zasadach komisowych. W tym drugim przypadku problem odpowiedzialności za usterki będzie sprawą umowy między autorem programu i nabywcą.

Ważnym zagadnieniem, które jedynie sygnalizujemy, jest stworzenie standardów dokumentacji programów i ich testowania. Specyfika programów mikroprocesorowych, wyrażająca się m.in. w dużych różnicach wielkości programów, które mogą być traktowane jako produkt handlowy (od kilkudziesięciu do kilkudziesięciu tysięcy bajtów) — powoduje dodatkowe kłopoty. Trudno bowiem wyobrazić sobie, aby ten sam standard dokumentacji i testowania obowiązywał w obu przypadkach.

Niektóre programy, m.in. translator języka BASIC, są udostępniane bezpłatnie w ramach akwizycji i reklamy.

Zagadnieniami perspektywicznych zastosowań mikroprocesorów zajmowano się dotychczas głównie przez odpowiednie ukierunkowanie współpracy z innymi placówkami naukowo-badawczymi. Przykładowo, w Instytucie Informatyki Uniwersytetu Warszawskiego prowadzono w ramach umowy prace teoretyczne dotyczące systemów wieloprocesorowych i ich mikroprocesorowych realizacji, a także — dotyczące synchronizacji procesów współbieżnych i jej wyrażenia w językach programowania [3] W Instytucie Maszyn Matematycznych zrealizowano translator skrośny języka MODULA [16]. Przewiduje się wykorzystanie tych prac w zastosowaniach przemysłowych (np. w wieloprocesorowym systemie sterowania w centralach telefonicznych), a także — przez samych użytkowników (np. — translatora języka MODULA) za pośrednictwem biblioteki B.ITE.M.

SZKOLENIE UŻYTKOWNIKÓW

W celu zrealizowania trzeciego zadania, we współpracy z Centrum Szkoleń Informatycznych ZETO w Łodzi, **zorganizowano cykl kursów dla potencjalnych użytkowników. Tematyka tych kursów dotyczyła zagadnień architektury systemów mikroprocesorowych, programowania w języku symbolicznym oraz w języku PL/M-80, a także — zastosowań metody modelowania strukturalnego.** Dla uczestników opracowano materiały pomocnicze w postaci skryptów, a dla wykładowców — folie i przezrocza.

W latach 1978—1980 w Centrum Szkoleń Informatycznych w Łodzi przeprowadzono 15 kursów, na których przeszkolono około 600 osób z 74 zakładów pracy i wyższych uczelni z całego kraju. W 1981 r. odbyło się sześć kursów z udziałem ok. 250 osób. W latach 1980—1982 odbywały się również cykliczne seminaria pn. Mikroprocesorowa Szkoła Zimowa. Dotychczas przeprowadzono je trzykrotnie, każdorazowo z udziałem kilkudziesięciu osób. Tematem III Szkoły Mikroprocesorowej, która odbyła się w maju 1982 r., były języki programowania wysokiego poziomu dla mikroprocesorów, systemy uruchomieniowe oraz oprogramowanie skrośne dla mikrokomputerów jednoukładowych typu INTEL 8048/8041.

Założeniem programowym Szkół Mikroprocesorowych jest stworzenie forum wymiany informacji o wykonanych pracach i zapotrzebowaniu na prace z zakresu systemów mikroprocesorowych — między osobami zaawansowanymi w tej dziedzinie i nowicjuszami. Dotychczasowe doświadczenia wydają się wskazywać, że zastosowana formuła organizacyjna jest właściwa.

Popularyzacji wiedzy służy również wykład nt. mikroprocesorów, ich programowania i zastosowań, prowadzony od 1979 r. w ramach Kursów Zastosowań Matematyki, organizowanych przez Instytut Matematyczny PAN, Instytut Podstaw Informatyki PAN oraz Wydział Matematyki, Informatyki i Mechaniki UW. Do chwili obecnej wykład ten prowadzono siedmiokrotnie, każdorazowo dla ok. 120 słuchaczy.

* * *

Problemy oprogramowania systemów mikroprocesorowych mają wiele cech wspólnych z problemami programowania w „klasycznej" informatyce, ale są też istotne różnice. Jedną z najważniejszych — zdaniem autora — jest skala potrzeb.

Jeśli mikroprocesory mają spełnić rewolucyjną rolę w procesach innowacyjnych w technice, produkcji, organizacji, dydaktyce — to rozpowszechnienie umiejętności programowania musi przewyższyć co najmniej dziesięciokrotnie stan obecny. Aby to osiągnąć, należy dokonać istotnych zmian w procesach nauczania.

Spożytkowanie wszystkich korzyści, jakie może przynieść rewolucja mikroprocesorowa, jest uwarunkowane w skomplikowany sposób, m.in. oporami społecznymi i reakcjami obronnymi na nowości. Wycinek tego problemu rozpatrzono w artykule [7], podejmując próbę sformułowania konkretnych wskazówek dla tych, którzy chcieliby poważnie potraktować działalność w dziedzinie zastosowań systemów mikroprocesorowych — z uwzględnieniem przynajmniej ważniejszych trudności, na jakie mogą się natknąć.

LITERATURA

[1] Ciok A.: Simulation of Billiards in SIMULA. Simula Newsletters, Vol. 6, No. 4, November 1978
[2] Ciok A.: Telewizyjna gra w bilard — algorytmy i realizacja. Prace Instytutu Technologii Elektronowej, Zeszyt 14, s. 21—26, 1978
[3] Czaja L., Doboslewicz W. i in.: Sprawozdania z realizacji umowy ITE/N-31/27/78 — Rz/24/78, Część I, II, III. Instytut Informatyki UW, Warszawa, 1978—1980
[4] Dańda J.: SIMULA and Structured Modelling. SIMULA Newsletters, Vol. 6, No. 4, November 1978
[5] Dańda J.: Informatyczne wspomaganie realizacji oraz interpretacji kwestionariuszy osobowości. Mikroprocesorowa Szkoła Zimowa, Łódź 1980; Materiały ZOINTE ITE NPCP, Warszawa, 1980
[6] Dańda J.: Sieć B.ITE.M — Biblioteki Programów Mikroprocesorowych. Koncepcja — pierwsze wyniki — problemy. Mikroprocesorowa Szkoła Zimowa, Łódź, 1980; Materiały ZOINTE ITE NPCP, Warszawa, 1980
[7] Dańda J.: Sposób organizowania działań zastosowaniowych w mikroinformatyce. Mikroprocesorowa Szkoła Zimowa, Łódź, 1981; Materiały ZOINTE ITE NPCP, Warszawa, 1981
[8] Grandys A., Namysłowski L.: Koncepcja systemu automatyzującego prace administracyjno-edycyjne biblioteki programów mikroprocesorowych B.ITE.M. Mikroprocesorowa Szkoła Zimowa, Łódź, 1980; Materiały, ZOINTE ITE NPCP, Warszawa, 1980
[9] Grandys A., Namysłowski L.: PL/M-80. I Opis języka. II Programowanie. III Zbiór ćwiczeń. Skrypt CSI ZETO, Łódź, 1980
[10] Myers G. J.: Projektowanie niezawodnego oprogramowania. WNT, Warszawa, 1980
[11] Oktaba H., Ratajczak W.: SIMULA 67. WNT, Warszawa, 1980
[12] Pluta A., Pluta E.: Konwersacyjne wykonywanie kwestionariuszy osobowości Eysencka i Cattella z wykorzystaniem mikroprocesorowego systemu typu MDS. Mikroprocesorowa Szkoła Zimowa, Łódź, 1980; Materiały, ZOINTE ITE NPCP, Warszawa, 1980
[13] Poznański Z.: SIMLIB — pakiet programów do symulacji cyfrowej na mikroprocesorze INTEL 8080. Prace Instytutu Technologii Elektronowej, Zeszyt 6, s. 49—67 (1980)
[14] Poznański Z.: SIMLIB — SIMULA-based Simulation Package for Microcomputer System Design. SIMULA Newsletters, Vol. 9, No. 1, February 1981
[15] Poznański Z., Szynka J.: Oprogramowanie mikroprocesorowego sterownika pieca dyfuzyjnego metodą modelowania strukturalizowanego. Prace Instytutu Technologii Elektronowej, Zeszyt 6, s. 33—48 (1980)
[16] Witaszek J.: Język programowania MODULA 8080 i jego kompilator skrośny. Materiały III Szkoły Mikroprocesorowej, CSI ZETO, Łódź, 1982
[17] Założenia ekonomiczno-organizacyjne funkcjonowania systemu B.ITE.M. Projekt. ZETO, Łódź, 1981.

ANTONI KRECZMAR
ANDRZEJ SALWICKI
Instytut Informatyki
Uniwersytet Warszawski

Język programowania LOGLAN. Część 3

Poniższy artykuł kończy trzyczęściowy cykl o LOGLANIE, rozpoczęty w numerze 7/82. W następnych, tegorocznych numerach INFORMATYKI zamieszczone zostaną dalsze artykuły o języku LOGLAN, zajmujące się takimi związanymi z nim pojęciami, jak: klasy, prefiksowanie czy programowanie współbieżne. (Red.)

MECHANIZMY OCHRONY

Mechanizmy ochrony ułatwiają bezpieczne programowanie. Jeżeli program jest duży, używa klas systemowych lub pisany jest przez grupę programistów, to niebagatelną sprawą jest tworzenie pewnych barier i zakazów dotyczących stosowania atrybutów.

Wyobraźmy sobie pewną strukturę danych, zdefiniowaną jako klasa. Niektóre z atrybutów tej klasy są dostępne dla użytkowników tej klasy, inne są lokalne i nie powinny być dostępne dla niepowołanych. Przykładowo, dla struktury *stos* (przedstawionej w poprzedniej części artykułu), takie atrybuty jak *pop* i *push* powinny być dostępne dla użytkowników tej klasy, natomiast takie atrybuty jak *top* czy *A* — nie powinny być dostępne. Niewłaściwe użycie lokalnego atrybutu może grozić niepoprawnym działaniem struktury danych.

Aby zabronić używania takich atrybutów, wprowadziliśmy do języka trzy mechanizmy zastrzeżeń: *close*, *hidden* i *taken*.

Zastrzeżenie *close* zdefiniowane wewnątrz klasy zabrania zdalnego dostępu do atrybutów klasy (na zewnątrz tej klasy). Na przykład:

```
unit A: class;
  var x: integer;
  close x,y,z;
  var y,z real;
  ....
end
```

oznacza, że zdalny dostęp z zewnątrz klasy *A* do atrybutów *x*, *y*, *z* jest zabroniony.

Zastrzeżenie *hidden* (syntetycznie zdefiniowane podobnie) nie zezwala na użycie takich atrybutów nie tylko przez zdalny dostęp, ale również w klasach prefiksowanych daną klasą. Atrybut taki może być użyty jedynie w samej klasie.

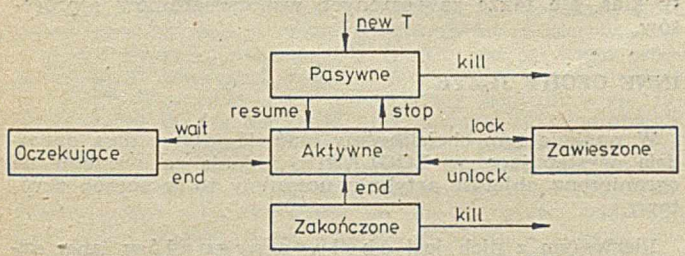
Zastrzeżenie *taken* określa atrybuty pobierane z prefiksu. Rozważmy moduł *B* prefiksowany klasą *A*. W module *B* możemy zdefiniować na liście *taken* te atrybuty klasy *A*, których chcemy używać w module *B*. Zastrzeżenie to jest w pewnym sensie dualne względem zastrzeżenia *hidden* i chroni użytkownika klasy *A* przed nieopatrznym (tzn. mimowolnym) użyciem atrybutu z prefiksu *A*. Oczywiście, jeżeli lista *taken* nie występuje w module *B*, to wszystkie dostępne atrybuty z klasy *A* są pobierane do modułu *B*.

OBLICZENIA RÓWNOLEGŁE

Pojęcie procesów działających równolegle stanowi wzbogacenie pojęcia współprogramów. Przypomnijmy, że współprogramy są obiektami, które mogą działać niezależnie, podobnie jak procesy. System współprogramów ma dwie operacje: *attach* i *detach*. Powodują one, że procesor uaktywnia odpowiedni obiekt, zawieszając obliczenia w obiekcie dotąd aktywnym, tzn. w danym momencie co najwyżej jeden współprogram może być aktywny. Odmienna sytuacja ma miejsce w systemie procesów. Podstawowa różnica polega na tym, że w programie można przekazać sterowanie do procesu, nie odbierając jednocześnie sterowania procesowi aktywnemu.

Tworzenie obiektu typu *process* polega na wykonaniu operacji *new process*. Proces nowoutworzony jest przerywany. Obudzenie (uaktywnienie) procesu *A* ma miejsce w wyniku wykonania instrukcji *resume (A)*.

Głównym problemem programowania z użyciem procesów jest synchronizacja. Dlatego wprowadzamy w języku LOGLAN nowy typ pierwotny *semaphore* oraz pewną liczbę operacji elementarnych działających na zmiennych tego typu. Zmienna typu *semaphore* synchronizuje wejście do ciągu instrukcji, które mogą być wykonywane tylko przez jeden proces. Ciąg taki nazywa się regionem krytycznym. Zmienna typu *semaphore* może być w dwóch stanach: *true* (semafor podniesiony) lub *false* (semafor opuszczony). Proces aktywny może wykonywać swoje wewnętrzne instrukcje, bądź też jedną z instrukcji zmieniających stan procesu (rys.):



- Instrukcja stop przenosi proces w stan PASYWNY.

- Instrukcja lock (semafor) w zależności od stanu semafora: opuszcza semafor i pozwala procesowi przejść do instrukcji następnej, albo — w przypadku gdy semafor jest opuszczony — powoduje przejście procesu w stan ZAWIESZONY.

- Instrukcja *unlock* (semafor) powoduje, że proces, który tę instrukcję wykonał przechodzi do wykonania następnej instrukcji, a pewien proces oczekujący na zwolnienie semafora będącego parametrem będzie uaktywniony.

- Bezparametrowa funkcja *wait* powoduje zawieszenie obliczeń w procesie do momentu zakończenia obliczeń (*end* procesu) przez jakiegokolwiek jego syna. Nazwa procesu zakończonego przekazywana jest jako wartość funkcji *wait*, a proces oczekujący wznawia działanie.

Przedstawiony powyżej system procesów i semaforów na pewno nie jest strukturalny, nie jest on zalecany jako język dla użytkowników. Posługując się mechanizmami klas i prefiksowania można w LOGLANIE zdefiniować wiele nowych mechanizmów synchronizacji i języków dla obliczeń równoległych. Poniżej przytoczony zostanie przykład wskazujący jak w LOGLANIE można zaimplementować pojęcie monitora [2]. Wspomnijmy, że w pracy T. Müldnera [3] znaleźć można system implementujący zadanie *(task)* i mechanizm podawania rąk *(hand shaking)* zaproponowany dla ADY. Użytkownik może zdefiniować inne własne systemy pracy procesów.

Monitor jest to struktura danych organizująca i synchronizująca dostęp do wspólnych zasobów. Rodzaj i ilość tych zasobów definiuje sam użytkownik, monitor ma za zadanie zrealizować jedynie bezkonfliktowy do nich dostęp. Bezkonfliktowość dostępu do wspólnych zasobów monitora realizuje wywołanie tzw. procedury wejściowej. Wejściowa procedura ma specjalny systemowy prefiks (*entry*), który organizuje właśnie wyłączność dostępu do zasobów wspólnych. Ponieważ użytkownik musi mieć także możliwość zawieszenia procesów wchodzących do monitora i szeregowania ich działania, monitor daje do dyspozycji użytkownika operacje na wewnętrznych kolejkach. Elementami takich kolejek są procesy, które weszły do monitora, a których działanie użytkownik zawiesił. Monitor dostarcza dwie procedury operujące na takich kolejkach:

delay (Q) — zawiesza działanie procesu, wstawia go do kolejki *Q* i otwiera dostęp do monitora

continue (Q) — uruchamia działanie pierwszego procesu z kolejki *Q*, o ile nie była pusta, w przeciwnym przypadku — otwiera dostęp do monitora.

Deklaracja klasy *Monitor* jest następująca:

```
unit Monitor: class;
  hidden sem,queue;
  var sem:semaphore; (* sem — semafor stojący na wejś-
                                          ciu do monitora *)
unit queue: class; (* kolejka dowolnych procesów *)
var front,tail: elem;
hidden front, tail,elem;
  unit elem:class (P:process);
    var succ: elem;
  end elem;
unit into: procedure (P:process);
var q: elem;
begin
  q: = new elem (P);
    if tail=none
      then
        front,tail:=q
      else
        tail.succ:=q; tail:=q;
      fi
    end into;
    unit out: function: process;
    var q: elem;
    begin
      if front =/= none
      then
        result:=front.P;
        q:=front.succ;
        kill (front);
        front:=q
      fi;
    end out;
  unit empty: function: boolean;
  begin
  result:=front=/=none
  end empty;
end queue;
unit entry: class;
var busy: boolean;
(*busy=true jeżeli wykonano continue do niepustej ko-
    lejki *)
hidden busy;
  unit delay: procedure (Q:queue);
  begin
    call Q.into (this process); (* wstaw aktywny proces do
      kolejki *)
      stop (sem)        (* otwór dostęp do monitora *)
  end delay;
  unit continue: procedure (Q:queue);
  begin
    if not Q. empty
    then
      busy:=true
        resume (Q.out)        (*uaktywnij kolejny proces *)
    fi;
  end continue;
```

```
begin      (* początek instrukcji klasy entry *)
  lock(sem);      (* wejście do regionu krytycznego *)
  inner;      (* treść procedury wejściowej *)
  if not busy then unlock (sem) fi;
  end entry;
end Monitor;
```

Prosty przykład skrzynki kontaktowej z buforem cyklicznym może być opisany jako klasa prefiksowana pojęciem monitora:

```
unit Mailbox: Monitor class (type T; size: integer);
  var pool: arrayof T,
    count, in_index, out_index: integer,
    readers_queue, writers_queue: queue;
  unit writer: entry procedure (r:T);
  begin
    if count = size then call delay (writers_queue) fi;
    in_index:=in_index mod size +1 ; count:=count+1;
    pool (in_index) :=r; call continue (readers_queue)
  end writer;
  unit reader: entry procedure (out r:T);
  begin
    if count = 0 then call delay (readers queue) fi;
    out_index:=out_index mod size +1;  count:=count + 1;
    r:=pool(out_index); call continue (writers_queue)
  end reader;
begin
  array pool dim (1:size);
  readers_queue:= new queue;
  writers_queue:= new queue;
end Mailbox;
```

PROGRAMOWANE ODŚMIECANIE

Programowane odśmiecanie stanowi ważną cechę semantyczną LOGLANU. Odśmiecanie polega na usuwaniu zbędnego pola z pamięci komputera. Na przykład — wywołanie procedury lub funkcji, a także otwarcie nowego bloku, wymaga najpierw zarezerwowania nowego pola w pamięci komputera na atrybuty lokalne takiego modułu. Po wykonaniu czynności przewidzianych przez ciąg instrukcji modułu, pole to może być natychmiast zwolnione i następnie użyte do innych celów. To jest właśnie przykład odśmiecania.

W językach implementowalnych stosowo (jak np. ALGOL-60), pola modułów są rezerwowane i zwalniane w sposób stosowy. Niestety, w językach z rekordami dynamicznymi już tak zrobić nie można (por. implementację PASCALA, gdzie na rekordy dynamiczne przeznacza się całkiem inny obszar pamięci, niż na pola modułów). W LOGLANIE (a także już w SIMULI) sytuacja jeszcze bardziej się komplikuje, ponieważ obiekty klas, współprogramów i procesów nie mogą być usuwane tak jak obiekty procedur, funkcji i bloków. Decyzję o usunięciu takiego obiektu może podjąć sam programista lub też można ją przerzucić na procedurę zwaną odśmiecaczem (ang. *garbage collector*). Odśmiecacz przegląda w pewnych momentach wykonywania programu wszystkie obiekty i usuwa te, do których program i tak już nie ma dostępu. Pierwsza metoda nazywa się odśmiecaniem programowanym, druga — odśmiecaniem automatycznym.

W języku LOGLAN z wielu powodów przyjęto możliwość programowanego odśmiecania. Jest to metoda znacznie bardziej efektywna (w szczególności przy równoległości działań), a także nie wyklucza możliwości automatycznego odśmiecania. Ponadto programowane odśmiecanie umożliwia oszczędniejszą gospodarkę pamięcią. Użytkownik zwalnia natychmiast zbędne pola danych i nie musi czekać na ewentualne odśmiecenie całej pamięci.

Syntaktycznie instrukcja powodująca programowane odśmiecanie ma postać:

```
kill (X)
```

gdzie X jest zmienną referencyjną. Po wykonaniu tej instrukcji obiekt wskazany przez X jest usuwany, a wszystkie zmienne, które wskazują na ten obiekt, przyjmują wartość *none*. Na przykład — jeżeli X, Y wskazują na ten sam obiekt klasy *A*, o atrybucie — powiedzmy — *a*, to po wykonaniu:

```
kill (X) ;
```

dostęp zdalny do atrybutu *a* w instrukcji:

```
Y.a
```

będzie niepoprawny, gdyż wartością Y jest *none* (sygnalizowany jest błąd w trakcie wykonania programu).

Właściwa, tzn. bezpieczna realizacja programowanego odśmiecania wymaga szybkich i niebanalnych algorytmów zarządzania pamięcią. Algorytmy takie zostały zaprojektowane specjalnie na potrzeby LOGLANU. Podkreślamy tę sprawę z tego względu, że w wielu językach programowania (np. PASCAL, ADA) wprowadzono inne, „niebezpieczne" programowane odśmiecanie. Semantykę takiego niebezpiecznego odśmiecania można zilustrować następująco. Rozważmy operację *kill (X)* i — tak jak poprzednio — niech zmienna Y wskazuje na to samo pole co X. Zdalny dostęp postaci *Y.a* do atrybutu nieistniejącego pola nie powoduje sygnalizacji błędu w trakcie wykonania programu (zjawisko to w terminologi angielskiej nazywa się *dangling reference*). Fatalne konsekwencje takiej niebezpiecznej semantyki Czytelnik może sobie sam wyobrazić.

Za pomocą operacji *kill* można usuwać nie tylko obiekty klas, ale także zawieszonych współprogramów i procesów.

INNE CECHY JĘZYKA

W punkcie tym chcielibyśmy omówić jeszcze dwa narzędzia występujące w LOGLANIE, jednak ze względu na ograniczoną objętość artykułu uczynimy to w sposób skrótowy.

Pierwszym z nich jest obsługa sygnałów (ang. *exception handling*), zwana także obsługą wyjątków, która polega na uwzględnianiu w programie sytuacji wyjątkowych. Sytuacja wyjątkowa powstaje wtedy, gdy w trakcie wykonywania programu pojawi się błąd (np.: dzielenie przez zero, przekroczenie zakresu indeksu zmiennej indeksowej, próba zdalnego dostępu do nieistniejącego pola), przerywając normalne wykonywanie programu. Sytuacja wyjątkowa pojawia się także wtedy gdy sam programista, stosując specjalną instrukcję, przerywa normalne wykonywanie programu.

Obsługa sygnałów pozwala przewidzieć i obsłużyć w programie takie sytuacje wyjątkowe. Obsługa sygnałów kosztuje, ale tylko wtedy, gdy wystąpi wysłanie sygnału. Jeżeli sygnał nie pojawi się, to program pracuje normalnie i to znacznie szybciej, niż gdybyśmy wszystkie sytuacje wyjątkowe uwzględniali na zapas, za pomocą stosowanych instrukcji warunkowych. Jeżeli „wyjątek" nie jest wyjątkowy, należy taką sytuację uwzględnić w sposób normalny, bez wprowadzania *handlera*.

Drugim ze wspomnianych narzędzi jest oddzielna kompilacja. Otóż uniwersalny język programowania powinien ułatwiać programowanie i uruchamianie dużych programów. Wymaga to od projektantów języka uwzględnienia możliwości oddzielnej kompilacji. W LOGLANIE przewidujemy taką możliwość.

Jednostką kompilacyjną w języku LOGLAN może być dowolny moduł (procedura, klasa, współprogram itp.), nie odwołujący się do zewnętrznych deklaracji. Rozróżniamy dwa rodzaje jednostek kompilacyjnych:

— jednostki binarne, gotowe do wykonania
— jednostki biblioteczne.

Jednostka binarna jest to jednostka kompilacyjna, przygotowana do połączenia z innymi jednostkami kompilacyjnymi. W LOGLANIE przewidujemy zatem możliwość operowania na bibliotekach jednostek kompilacyjnych wprowadzając odpowiednie dyrektywy dla kompilatora.

Oddzielna kompilacja modułów w połączeniu z innymi możliwościami języka (jak np. prefiksowanie) stwarza niezwykle dogodne narzędzia do pracy nad dużymi programami i systemami. Z jednej strony zachowujemy tę możliwość, którą oferuje FORTRAN i asemblery — możliwość wymiany pojedynczych modułów. Z drugiej — mamy możliwość tworzenia już nie tylko bibliotek procedur, lecz całych języków problemowo-zorientowanych, które mogą być przy tym zorganizowane w strukturę hierarchiczną.

* * *

Przedstawiliśmy cechy charakterystyczne języka LOGLAN. Czytelnik zapewne zgodzi się z nami, że wiele narzędzi programistycznych oferowanych przez LOGLAN pojawia się po raz pierwszy w uniwersalnym języku programowania lub też po raz pierwszy można z takiego narzędzia bez kłopotu korzystać w Polsce.

Weźmy pod uwagę współprogramy — narzędzie to jest dostępne użytkownikom SIMULI, ale w Polsce mamy dwie instalacje SIMULI 67, obie nieudane, w systemach CYFRONET. Obsługa wyjątków w postaci zalążkowej występuje w PL/I, ale nieznane nam są żadne większe produkty programowe korzystające z tego narzędzia. Pełniejszy system obsługi wyjątków występuje w ADZIE, lecz kompilator ADY ma powstać dopiero w 1983 r. Obsługa wyjątków w LOGLANIE jest mechanizmem znacznie bardziej rozwiniętym, bardziej funkcjonalnym. Podobnie rzecz się ma i z innymi narzędziami LOGLANOWYMI.

Chcemy powtórzyć pewne metodologiczne aspekty, które przedstawiamy pod rozwagę programistów i kierowników zespołów programistycznych. LOGLAN jest językiem umożliwiającym tworzenie dużych produktów programowych w sposób zdyscyplinowany i poddający kontroli wiele założeń, które do tej pory nie były zawarte w programach, a jedynie w dokumentacji i umysłach programistów. Porównajmy — dwadzieścia lat temu niewiele komputerów miało wbudowaną arytmetykę zmiennoprzecinkową. Programista był odpowiedzialny za to, by przed dodaniem dwu liczb zapisanych w różnych skalach wyrównać pozycję przecinka. Drobny to fakt, lecz ileż pociągnął błędów programistycznych. Podobnie programowanie w języku adresów symbolicznych i programowanie w językach wyższego poziomu (powstałych w latach sześćdziesiątych) pozwala dostrzec wyraźne zalety pisania programów w taki sposób, który na maszynę przerzuca większość pracy nietwórczej, automatycznej i pozostawia programiście więcej czasu na skupienie się nad istotą algorytmu.

Praktyka ostatnich lat wykazała wszystkie niedogodności takich języków, jak FORTRAN, ALGOL czy PASCAL. W naturalny sposób pojawiają się próby uczynienia następnego kroku. Główne problemy, jakie muszą rozwiązać konstruktorzy nowych języków, to stworzenie dobrych narzędzi do opisywania struktur danych, obliczeń współbieżnych i pracy dużych zespołów programistycznych. LOGLAN jest próbą rozwiązania tych problemów. Nie twierdzimy, że jest to ostatnie słowo w dziedzinie projektowania języków. Prawdę mówiąc — aż do chwili, gdy zaczęliśmy uruchamiać własne programy, nie mieliśmy pewności, czy język ten nada się do praktycznego programowania. Znane nam są przykłady pięknych idei, które okazały się niewypałem (np. ALGOL 68). Z upływem miesięcy przekonaliśmy się, że można w LOGLANIE opisywać bardzo duże systemy i uzyskiwać jednocześnie dużą ich efektywność. Wiele przykładów przekonało nas, że w LOGLANIE można szybko zaprogramować i uruchomić programy, dla których odpowiedniki FORTRANOWE czy ALGOLOWE po prostu nie istnieją.

„Filozofia" LOGLANU to rozwijanie oprogramowania w naturalnych terminach zadania i warstwowa praca nad jego implementacją. Nietrudno napisać program, który modeluje system bankowy, przemysłowy czy telekomunikacyjny. Można pracę tworzenia modułów odpowiadających różnym składnikom systemu podzielić pomiędzy różnych programistów. Umożliwia to przejrzysta i modularna struktura języka; twórca logarytmu jest w stanie posługiwać się pojęciami na pewnym poziomie abstrakcji, nie troszcząc się o ich realizację. Pojęcia te — zdefiniowane przez innego programistę lub wzięte z biblioteki systemowej — mogą zostać użyte w sposób bezpośredni dzięki mechanizmowi prefiksowania. Takie podejście zmniejsza redundację tekstu programu oraz pracy programistów.

LOGLAN dostarcza bogatych narzędzi dla bezpiecznego rozwijania dużych systemów. Przepływ informacji pomiędzy zesołami programistów jest ułatwiony dzięki czytelności zapisu, a poszczególne części oprogramowania dają się w łatwy sposób modyfikować. Dzięki właściwie stosowanym mechnizmom ochrony, wprowadzane zmiany w jednym module nie powodują nieoczekiwanych efektów ubocznych w innych miejscach programu. Stwarza to również dobre warunki do systematycznego testowania programu, będącego jedną z ważnych części jego weryfikacji. Wiele z błędów, które w FORTRANIE możliwe były do wykrycia dopiero w trakcie wykonania programu, w LOGLANIE sygnalizuje się już na etapie kompilacji.

Podobną ideologię wyznaje ADA. Nie podejmujemy się w tym miejscu wytłumaczyć wszystkich podobieństw i różnic pomiędzy ADĄ a LOGLANEM. Chcielibyśmy stwierdzić, że w założeniach LOGLANU jest dynamika i choć fakt ten skomplikował prace implementacyjne, to podporządkowaliśmy mu wszystko. W ADZIE dynamika pojawia się tylko dwa razy i w obu przypadkach nastręcza wiele trudności użytkownikom i przyszłym implementatorom: tworzenie wielu obiektów wg deklaracji typu *(access type)* i procesy współbieżne *(tasking)* nie pasują do reszty języka; do dziś nie jest znana precyzyjna i jednoznaczna semantyka tych pojęć. Pakiety, które w ADZIE mają umożliwić tworzenie języków problemowo zorientowanych, nie spełniają wszystkich oczekiwań. Pełniejsze krytyczne omówienie ADY i LOGLANU można znaleźć gdzie indziej [1, 3].

Prace nad LOGLANEM nadal trwają. Przewidujemy wprowadzenie dodatkowych narzędzi pierwotnych do pracy z plikami. W oparciu o nie będzie można stworzyć klasy — języki do obsługi plików różnych rodzajów. Wprowadzone będą tablice statyczne i rekordy. W obecnej wersji kompilator nie realizuje oddzielnej kompilacji modułów. Można jednak w zaskakująco atrakcyjny sposób wykorzystać kompilator LOGLANU dla wzbogacenia języka — tak wprowadziliśmy równoległość do LOGLANU.

Najważniejsze dla przyszłości LOGLANU będą próby tworzenia dużych systemów. Przeprowadzono już wiele udanych eksperymentów w dziedzinie symulacji, w programowaniu współbieżnym.

Zachęcamy Czytelnika do sprawdzenia, czy nasze oczekiwania są realne. Sprawdzianem LOGLANU będą m.in. napisanie kompilatora LOGLANU (a może i ADY) w LOGLANIE, stworzenie narzędzi dla prac z bazami danych oraz zbudowanie paru systemów czasu rzeczywistego.

LITERATURA

[1] Cioni G., Kreczmar A.: Critical remarks on the language ADA. Universita di Roma report, 1980

[2] Hoare C. A. R.: Monitors, an operating system structuring concept, CACM, vol. 17, 10, October 1974, 549—557

[3] Müldner T.: Pewne uwagi o dwóch nowych językach wysokiego poziomu LOGLAN i ADA. Biuletyn Techniczny MERA nr 11—12/1980.

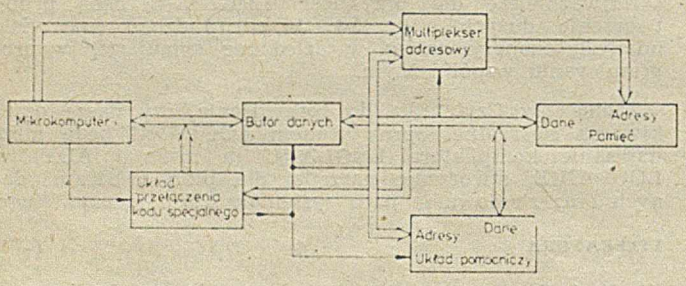
Zwiększenie efektywności mikrokomputera

Mikrokomputery, jako urządzenia scalone realizują rozkazy nie podlegające zewnętrznej modyfikacji. Jeżeli chcemy wykonać potrzebną operację, a nie ma odpowiedniego rozkazu języka wewnętrznego, to stosujemy zazwyczaj pewną sekwencję rozkazów stanowiących na ogół podprogram. Przykładem może być realizacja operacji mnożenia dla mikroprocesora INTEL 8085. Nie jest to jednak rozwiązanie korzystne, bowiem czas wykonania podprogramu jest dość długi.

Jeżeli zależy nam na zwiększeniu szybkości wykonywania operacji nie objętych listą rozkazów mikroprocesora, to zamiast rozwiązania programowego stosujemy na ogół rozwiązanie sprzętowe. Do mikroprocesora dołącza się dodatkowe, zewnętrzne układy pomocnicze, które — sterowane przez mikroprocesor — wykonują żądane operacje. Argumenty operacji są przesyłane z pamięci do mikroprocesora, a następnie do układu pomocniczego. Po wykonaniu operacji, mikroprocesor odczytuje wynik i odsyła go do pamięci. Stosuje się przy tym typowe techniki wejścia-wyjścia. Przykładem takiego rozwiązania może być dołączanie układu kalkulatorowego do systemu mikroprocesorowego.

Takie postępowanie nie należy jednak do najbardziej efektywnych, ponieważ mikroprocesor jest zaangażowany we wszystkie operacje przesyłania argumentów i wyników. Istnieje metoda lepszego rozwiązania tego problemu. Polega ona na zastosowaniu specjalnych, szybkich układów pomocniczych, które bez udziału mikroprocesora wykonują żądane operacje specjalne. Metoda ta, zwana metodą przeplatania, znacznie skraca czas wykonywania operacji specjalnych, tj. operacji nie objętych listą rozkazów mikroprocesora.

Zasadniczym elementem tej metody jest zmiana kodu całego rozkazu na inną postać, która uaktywnia szybkie układy zewnętrzne, a nie oddziałuje na mikroprocesor. Zwykle postać ta odpowiada kodowi rozkazu NOP, tj. „nic nie rób" lub kodowi skoku względnego JMP. Zamiana postaci kodu rozkazu zachodzi w fazie pobrania rozkazu. Pomocnicze układy zewnętrzne, połączone bezpośrednio z pamięcią systemu mikroprocesorowego, samodzielnie pobierają argumenty z pamięci oraz przesyłają do niej wyniki operacji.



Rys. 1. Ilustracja zasady mnożenia

Na rysunku 1 przedstawiono schemat blokowy systemu mikroprocesorowego, współpracującego z układami pomocniczymi według metody przeplatania. Układy pomocnicze są uaktywniane kodami rozkazów, tzw. kodami specjalnymi, na które mikroprocesor nie reaguje. Dopóki na szynie danych nie znajdzie się kod specjalny, zarówno multiplekser adresowy, jak i bufor danych — nie wprowadzają żadnych zmian w obiegu sygnałów w obrębie systemu mikroprocesorowego.

Jeżeli jednak, w fazie pobrania rozkazu, dekoder wykryje kod specjalny, to wysyła sygnał blokujący bufor danych oraz multiplekser adresowy. Dodatkowo uaktywnia się wtedy tzw. układ przełączania kodu specjalnego. Układ przełączania kodu specjalnego wysyła na szynę danych systemu mikroprocesorowego kod tzw. rozkazu pustego. Kod rozkazu pustego jest tak dobrany, aby po odebraniu go mikroprocesor nie wykonywał żadnego rozkazu (tak jak po odebraniu kodu rozkazu „nic nie rób"). Jednocześnie układy pomocnicze rozpoczynają własną pracę, niezależną od procesora. Pobierają argumenty z pamięci, wykonują operację i odsyłają wynik tej operacji do pamięci.

Jako bloki pomocnicze stosuje się układy o dużej szybkości działania, natomiast stosowane obecnie pamięci mają bardzo małe czasy dostępu, w porównaniu z czasami wykonywania rozkazów przez mikroprocesory. W związku z tym cały proces pobrania argumentów, wykonania operacji i odesłania wyniku do pamięci można zakończyć szybciej niż mikroprocesor wykona rozkaz pusty. Jeżeli jednak zachodzi obawa, że proces ten będzie trwał dłużej niż wykonywanie rozkazu pustego, to rozkaz pusty można rozszerzyć o kilka taktów oczekiwania. Zależy to oczywiście od typu stosowanego mikroprocesora oraz od rodzaju operacji wykonywanej przez blok pomocniczy.

Metodę przeplatania można stosować w przypadku każdego systemu mikroprocesorowego. Sposób rozwiązania zależy od typu mikroprocesora oraz od rodzaju operacji. Na ogół wykorzystuje się różne rozwiązania sprzętowe.

Rozważmy, przykładowo, mikrokomputer TMS 9900 współpracujący z zewnętrznym układem mnożącym TRW MPY-16. Mikrokomputer TMS 9900 zawiera w obrębie procesora jedynie trzy rejestry, tj.: licznik rozkazów (PC), rejestr stanu (ST) oraz rejestr obszaru roboczego (WH). W rejestrze obszaru roboczego przechowuje się adres komórki pamięci, w której znajduje się pierwszy z rejestrów roboczych. Pozostałe 15 rejestrów zajmują następne komórki pamięci.

Operacja mnożenia dla argumentów bez znaku, wykonywana programowo, trwa w przypadku mikrokomputera TMS 9900 ok. 20 μs. Zastosowanie metody przeplatania znacznie skraca ten czas — wykonywanie operacji mnożenia argumentów ze znakiem trwa wówczas nie dłużej niż 2 μs. Metoda przeplatania nie zmienia przy tym sposobu programowania mikrokomputera.

W przypadku mikrokomputera TMS 9900 rozkaz mnożenia ma następującą postać:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	1	1	0	D					TS	S			
Drugie słowo														(wyłącznie przy TS = 10)	

gdzie D jest numerem rejestru roboczego, w którym znajduje się argument, a TS i S — określają tryb adresowania argumentu w sposób podany poniżej.

Tabela 1

TS	S	Tryb adresowania
00	0...15	Rejestrowy obszaru roboczego
01	0...15	Pośredni rejestrowy obszaru roboczego
10[1])	0	Symboliczny
10[1])	1...15	Indeksowy
11	1...15	Pośredni rejestrowy obszaru roboczego z automodyfikacją

[1]) nie stosowany przy metodzie przeplatania

W symbolicznym i indeksowym trybie adresowania rozkaz składa się z dwóch słów, przy czym drugie słowo zawiera argument lub adres argumentu. W pozostałych trzech trybach adresowania cały rozkaz zajmuje wyłącznie jdeno słowo.

Tabela 2

Faza	Funkcja	Czas trwania [ns]
1	Pobranie rozkazu	
	(WH) + SRC → szyna adresowa	350
	(pamięć) → szyna danych	200
2	(szyna danych) → szyna adresowa (TS≠00)	50
3	(pamięć) → układ mnożący	200
4	(szyna adresowa) + 1 → pamięć (TS=11)	200
5	(WH) + DST → szyna adresowa	50
6	(pamięć) → układ mnożący	200
7	mnożenie	200
8	(bardziej znacząca część wyniku) → pamięć	200
9	(WH) + DST + 1 → szyna adresowa	50
10	(mniej znacząca część wyniku) → pamięć	200

[WH] — zawartość wskaźnika obszaru roboczego
SRC — adres źródła informacji
DST — adres odbiorcy informacji.

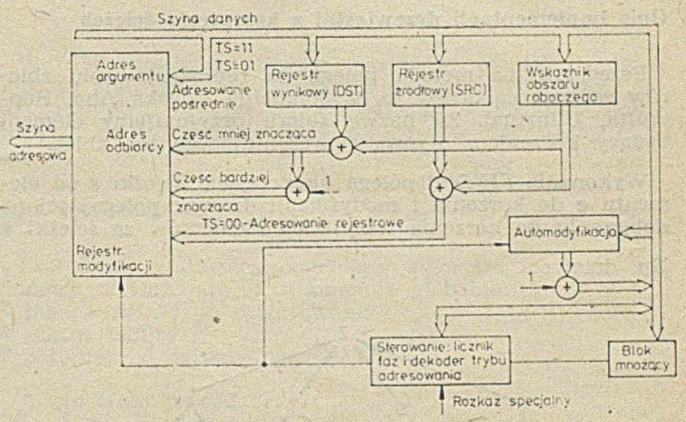
W rozpatrywanym przykładzie stosuje się właśnie te trzy tryby adresowania, a jeden z nich jest zarezerwowany wyłącznie do realizacji operacji mnożenia. Tryb symboliczny i indeksowy jest przeznaczony do wykonywania dwóch rozkazów pustych i właściwego ustawiania licznika rozkazów — tak, aby po wykonaniu rozkazu pustego procesor dalej realizował swój właściwy program.

Rozkaz mnożenia (001110) zostaje zdekodowany w fazie pobrania, w układzie przełączania rozkazu specjalnego. Układ ten blokuje bufor danych, a do mikrokomputera jest wysyłany kod 0000. Ponieważ mikrokomputer nie rozpoznaje tego kodu, nie wykonuje żadnego rozkazu i nie zmienia stanu.

Proces dekodowania rozkazu w mikrokomputerze kończy się po sześciu taktach pracy, po czym zawartość licznika rozkazów zwiększa się i procesor jest gotowy do pobrania następnego rozkazu. W tym samym czasie do układu mnożącego dochodzi kod rozkazu mnożenia. Tryb adresowania jest określony zawartością pola TS. Na tej podstawie układ mnożący pobiera adresy i argumenty z pamięci mikrokomputera. Może dokonać przy tym sześciokrotnego (TS = 11) dostępu do pamięci, aby pobrać żądane słowa i odesłać wynik operacji.

Sama operacja mnożenia, wykonywana w szybkim bloku mnożącym, trwa 200 ns, a cały proces — nie dłużej niż 2 μs. Przy czasie dostępu do pamięci równym 200 ns i czasie propagacji sygnałów w systemie równym 50 ns, proces można zakończyć w ciągu 1,9 μs.

W rozpatrywanym przykładzie zastosowanie metody przeplatania skraca czas wykonywania operacji mnożenia dziesięciokrotnie — w przypadku liczb bez znaku, a dwudziestokrotnie — w przypadku liczb ze znakiem. Dla porównania można podać, że zastosowanie zewnętrznych układów pomocniczych (np. układu kalkulatorowego) pracujących pod kontrolą mikroprocesora, skraca czas wykonywania operacji mnożenia odpowiednio dwukrotnie i czterokrotnie.



Rys. 2. Schemat blokowy systemu przeplatania

Układ mnożący rozpoczyna pracę po odebraniu sygnału wysłanego przez układ przełączania rozkazu specjalnego (rys. 2). Blok sterujący układu mnożącego dekoduje tryb adresowania, określony zawartością pól TS i S, oraz dekoduje zawartość pola D. Następuje aktualizacja zawartości rejestru obszaru roboczego i uruchomienie licznika faz. W rozpatrywanym przykładzie istnieje dziesięć faz, z tym że obecność dwu z nich, tj. fazy 2 i 4, zależy od trybu adresowania.

* * *

Opisana metoda umożliwia znaczne zwiększenie szybkości wykonywania żądanych operacji przez mikrokomputer. Można ją stosować w przypadku większości istniejących systemów mikroprocesorowych, bez konieczności wprowadzania zmian w wewnętrznej budowie tych systemów.

Oprac. MARIA WAŚNIEWSKA
na podstawie COMPUTER DESIGN
nr 2/1982

ALGORYTMY

Problem FIND-UNION

Omówimy poniżej problem znany w literaturze pod nazwą FIND-UNION (por. Aho A.V., Hopcroft J.E., Ullman J.D.: The Design and Analysis of Computer Algorithms, Addison-Wesley Pub.Co., 1976).

Załóżmy, że mamy daną rodzinę rozłącznych zbiorów. Elementy tych zbiorów oznaczać będziemy za pomocą kolejnych liczb naturalnych *1,2,...,n*. Określmy dwie operacje — *FIND* i *UNION:*

- wartością funkcji *FIND(e)* jest (jedyny!) zbiór *A*, którego *e* jest elementem
- po wykonaniu procedury *UNION (A,B,C)* — *C* jest równe sumie zbiorów *A* i *B*, natomiast *A* i *B* mają nieokreślone wartości.

Poniżej przedstawiamy bardzo efektywną implementację tych operacji.

Motywacja

Omawiana struktura danych ma liczne zastosowania. Przykładem może być algorytm Kruskala, w którym dla danego grafu buduje się drzewo rozpinające o najmniejszym

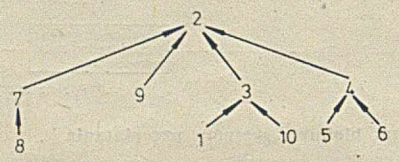
koszcie, czy też algorytm Hopcrofta-Karpa badania równoważności automatów skończonych. Dzięki umiejętnemu wykorzystaniu operacji *FIND* oraz *UNION* niektóre trudne algorytmy stają się przejrzyste, a ich implementacja — efektywna.

Opis implementacji drzewiastej z kompresją ścieżek

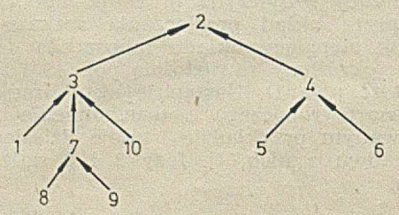
Przedstawiana metoda polega na reprezentowaniu zbiorów za pomocą drzew (p. wspomniana książka Aho, Hopcrofta, Ullmana). Za nazwę zbioru przyjmujemy element będący korzeniem drzewa reprezentującego ten zbiór.

Wykonanie *FIND(e)* polega na przejściu ścieżki *s* od elementu *e* do korzenia i modyfikacji drzewa, polegającej na dołączeniu do korzenia wszystkich elementów ze ścieżki *s*.

Np. drzewo:



po wykonaniu *FIND(9)* przyjmie postać:

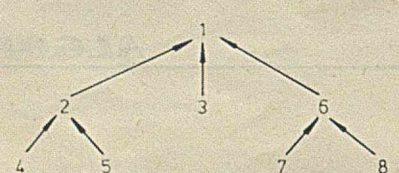


zaś wynikiem *FIND(9)* będzie *2*.

Wykonanie *UNION (A,B,C)* polega na dowiązaniu drzewa o mniejszej liczbie elementów do drzewa większego i przesłaniu drzewa wynikowego na zmienną *C*.

Jeśli — na przykład — *A* i *B* są następujące:

to, po wykonaniu *UNION (A,B,C)*, *C* przyjmie postać:



Ta implementacja jest zaskakująco efektywna. Niech bowiem *F* i *G* będą zdefiniowane następująco:

$$\begin{cases} F(0) = 1 \\ F(i) = 2^{F(i-1)} \text{ dla } i > 0 \end{cases}$$

$G(n) = \min \{k : F(k) \geqslant n\}$. Wówczas koszt wykonania *m* operacji *FIND* i *n* operacji *UNION* jest proporcjonalny do $(m+n)^* G(n)$. Zauważmy, że *G* rośnie bardzo wolno, np.

$G(n) \leqslant 5$ dla $n \leqslant 2^{65536}$.

Struktura danych

Jak już wspomnieliśmy, przyjmujemy dwa podstawowe założenia:

- rozpatrujemy rodzinę *F* zbiorów rozłącznych
- elementy zbiorów można rozpatrywać jako liczby całkowite *1,2,...,n*.

Początkowo $F = \{\{1\}, \{2\}, ..., \{n\}\}$. *F* reprezentujemy za pomocą dwóch tablic: *FATHER [1 : n]* oraz *COUNT [1 : n]*, przy czym

$$\text{FATHER[i]} = \begin{cases} \text{ojciec i-tego wierzchołka, jeśli i nie jest korzeniem} \\ 0, \text{ jeśli i jest korzeniem} \end{cases}$$

COUNT [i] = liczbie wierzchołków drzewa, jeśli i jest jego korzeniem

Zakładamy, że *n* oraz tablice *FATHER* i *COUNT* są globalne.

Opis procedur

Procedura *Init* inicjalizuje strukturę danych:

```
procedure Init;
begin integer i;
  for i:=1 step 1 until n do
  begin FATHER [i] :=0;
        COUNT [i] := 1
  end
end Init;
```

Parametry procedury *Union* są następujące:

A,B — liczby całkowite będące adresami korzeni drzew reprezentujących zbiory *A,B* w tablicy *FATHER*
C — zmienna całkowita wskazująca na korzeń drzewa wynikowego.

```
procedure Union (A,B,C) : integer A,B,C;
comment C: = A ∪B;
begin integer i ;
if FATHER [A] ≠ 0 or FATHER [B] ≠ 0 then goto error;
comment error jest etykietą obsługi błędu dla złych wartości A lub B;
if COUNT [A] > COUNT [B] then begin i :=A; A:=B;
   B:=i end;
FATHER [A] :=B; COUNT [B] := COUNT [A] + COUNT [B];
C :=B; A :=B :=O
end Union;
```

Funkcja *Find* (w książce Aho, Hopcrofta, Ullmana) wymagała dodatkowej pamięci na stos. Można ją jednak łatwo zmodyfikować w ten sposób, aby uniknąć strat pamięci (por. Banachowski L., Kreczmar A.: Elementy analizy algorytmów, WNT, Warszawa 1982).

Parametrem *Find* jest liczba naturalna *e* określająca element; wynikiem funkcji jest korzeń drzewa reprezentującego zbiór zawierający *e*.

```
integer procedure Find(e) ; integer e;
begin integer u,v,w;
  v:=e;
  while FATHER[v] ≠ 0 do v:= FATHER[v];
  comment teraz następuje kompresja ścieżki (v — korzeń drzewa);
  w:=e;
  while w ≠ v do
  begin u; = FATHER[w] ; FATHER[w]: =v; w:=u end
  Find :=v
end Find;
```

ANDRZEJ SZAŁAS
ZBIGNIEW ŚWIRSKI

BASIC

Zmodyfikowane sortowanie bąbelkowe

Opisana poniżej procedura sortowania wewnętrznego — modyfikacja sortowania bąbelkowego (por. W. Dobosiewicz: An Efficient Variation of Bubble Sort, Information Processing Letters, vol. 11, nr 1/1980) ma dwie podstawowe zalety: szybkość działania i krótki kod. Mamy więc nadzieję, że spotka się ona z zainteresowaniem użytkowników BASICU. Poniżej przedstawiamy tabelę czasów wykonania (w ms) uzyskanych na maszynie CDC CYBER 72 dla takich algorytmów, jak sortowanie bąbelkowe, sortowanie szybkie (QUICKSORT) oraz zmodyfikowane sortowanie bąbelkowe.

Czasy wykonania zadania sortowania elementów dla różnych algorytmów

Liczba sortowanych elementów	QUICKSORT [ms]	Sortowanie bąbelkowe [ms]	Zmodyfikowane sortowanie bąbelkowe [ms]
10	0,86	0,6	0,58
100	13,38	57,3	11,87
500	87,81	1456,6	84,80
1 000	199,8	–	194,9
10 000	2696,8	–	2743,8

Zachęcamy Czytelników również do porównania czasu działania omawianego algorytmu z czasem działania innych procedur sortujących.

Opis metody

Metoda algorytmu opiera się na spostrzeżeniu, że sortowanie bąbelkowe zachowuje się w sposób naturalny, działając tym szybciej, im ciąg wejściowy bliższy jest ciągowi posortowanemu. Można więc sortowanie podzielić na dwie fazy:

- częściowe posortowanie ciągu wejściowego
- dokończenie sortowania metodą bąbelkową.

Okazuje się, że do realizacji pierwszej fazy z powodzeniem nadaje się algorytm Shella (opis sortowania bąbelkowego oraz sortowania Shella można znaleźć np. w książce N. Wirtha: Algorytmy + struktury danych = programy, WNT, Warszawa 1980).

W celu określenia proporcji pomiędzy czasem działania obu faz algorytmu wprowadzamy parametr *T*. Wartość tego parametru wpływa na efektywność algorytmu. Nasze doświadczenia wykazały, że powinien zawierać się on między *2,8* a *3,1*.

Opis procedury

Procedura realizująca omawiany algorytm ma trzy parametry występujące jako zmienne globalne:

- tablica *A[1:N]* do posortowania
- zakres tablicy *N*
- parametr *T* opisany powyżej.

A oto treść procedury:

```
110 REM A — TABLICA OD 1 DO N, T — TRZECI PA-
    RAMETR
120 T1 = INT(T*LN(N))
130 H = INT(N/2)
140 FOR L=1 TO T1
150 FOR I=1 TO N—H
160 IF A (I) ⩽ A(I+H) THEN 200
170 X = A(I)
180 A(I) = A(I+H)
190 A(I+H) = X
200 NEXT I
210 H = INT(H*3/4)
220 NEXT L
230 L = N—1
240 IF L⩽0 THEN RETURN
250 K = 0
260 FOR I=1 TO L
270 IF A (I) ⩽ A(I+1) THEN 320
280 X = A(I)
290 A(I) = A(I+1)
300 A(I+1) = X
310 K = I
320 NEXT I
330 L = K—1
340 GO TO 240
```

ANDRZEJ SZAŁAS
ZBIGNIEW ŚWIRSKI

Specjaliści do spraw

W wielu przedsiębiorstwach państwowych powoływane jest stanowisko Głównego Specjalisty, najprościej byłoby powiedzieć — do spraw informatyki. Jednak spotyka się na ogół Głównego Specjalistę ds.:

— Systemów Zarządzania i Informatyki
— ETO
— Elektronicznego Przetwarzania Danych
— EMC
— Programowania i Informatyki
— Organizacji i Informatyki.

Pełni on najczęściej funkcje koordynatora prac projektowo-programowo-wdrożeniowych prowadzonych w przedsiębiorstwie, a także prac związanych z rozwojem zastosowań informatycznych w przedsiębiorstwie.

Nikt nie podważa zasadności nazw stanowisk Głównego Konstruktora, Głównego Technologa, Głównego Księgowego czy Głównego Ekonomisty. Stanowiska te mają określony zakres prac i obowiązków, dotyczą bowiem dziedzin tradycyjnie występujących w przedsiębiorstwie. Wprowadzanie informatyki powoduje natomiast często wątpliwości co do jej celów i zadań.

Można przypuszczać, że niezależnie od obecnego zróżnicowania nomenklatury stanowisk Głównych Specjalistów ds. Informatyki znakomita ich większość realizuje te same zadania. Jestem przekonana, że najodpowiedniejsze byłoby wprowadzenie stanowiska Głównego Informatyka w miejsce Głównych Specjalistów, co — poza wszystkim — ułatwiłoby jednoznaczne określenie miejsca informatyki w działalności przedsiębiorstwa.

ELŻBIETA KIERCZUK
Wrocław

Zastosowanie systemów informatycznych w szkołach wyższych

Rezultaty dotychczasowych prac projektowo-programowych zaspokajających potrzeby szkół wyższych — odpowiednio rozpowszechnione — mogą być wykorzystane w całym szkolnictwie wyższym, a także poza uczelniami — np. w biurach projektów lub ośrodkach badawczo-rozwojowych. W niniejszym artykule opisano rozwiązania informatyczne opracowane przez Politechnikę Wrocławską. Stanowią one zbiór niezależnie realizowanych tematów.

Opisane tu systemy informatyczne i pakiety programowe zostały przygotowane do eksploatacji na komputerach R-32, pracujących pod nadzorem systemu operacyjnego OS/JS. Zastosowano głównie wsadową technologię przetwarzania danych, a w niektórych przypadkach (np. w dziedzinie projektowania wspomaganego komputerowo) wykorzystano konwersacyjny dostęp do komputera. Do realizacji konwersacyjnego przetwarzania wykorzystano monitory ekranowe (typ EC 7917), pracujące w trybie lokalnym oraz podsystem podziału czasu TSO.

Przewiduje się, że niektóre z opisanych tematów mogą zostać przygotowane do eksploatacji w warunkach zdalnego dostępu do komputera, przy wykorzystaniu systemu teleprzetwarzania, którego głównym elementem jest procesor teleprzetwarzania danych (typ EC 8371.01).

SYSTEMY INFORMATYCZNE PRZEZNACZONE DO ROZPOWSZECHNIANIA

Poniżej przedstawiono szereg rozwiązań, dla których zakończone zostały prace projektowo-programowe i które są przygotowane do rozpowszechnienia.

Moduł oprogramowania podstawowego

Moduł oprogramowania podstawowego służy do przeprowadzania praktycznych ćwiczeń w ramach nauczania informatyki. Składa się on z dwóch części: pierwsza zaznajamia studentów z językiem opisu zadań systemu operacyjnego OS/JS, druga — zapoznaje ich z pracą w trybie wielodostępnym i sposobem wykorzystania podsystemu podziału czasu TSO.

Moduł ten przeznaczony jest zarówno dla studentów, którzy po raz pierwszy stykają się z systemem operacyjnym OS i nie posiadają prawie żadnego przygotowania z zakresu informatyki, jak i dla użytkowników bardziej doświadczonych.

System projektowania konstrukcji cyfrowych — DEDICO

System DEDICO służy do komputerowo wspomaganego projektowania urządzeń cyfrowych w zakresie projektowania montażowo-łączeniowego. W sferze dydaktyki system realizuje następujące funkcje:

- wspomaganie nauczania w zakresie metodologii i algorytmizacji projektowania konstrukcji cyfrowych; dotyczy to w szczególności zagadnień dekompozycji problemów i syntezy procesów projektowania oraz zastosowań różnych algorytmów projektowania występujących w tych procesach
- wspomaganie nauczania w zakresie ogólnej wiedzy, umiejętności i nawyków posługiwania się środkami informatyki w projektowaniu
- wspomagnie prac w zakresie projektowania konstrukcji cyfrowych: dotyczy to realizowanych w procesie dydaktycznym projektów, prac przejściowych i prac dyplomowych zawierających zagadnienia mieszczące się w zakresie systemu DEDICO.

Projektowanie konstrukcji metalowych

Pakiet programów z zakresu projektowania konstrukcji metalowych służy do komputerowego wspomagania projektowania zarówno prostych elementów konstrukcyjnych, jak i obiektów bardziej złożonych. Pakiet może być również stosowany w jednostkach projektowo-badawczych do kształcenia kadr inżynierskich w zakresie projektowania konstrukcji budowlanych, mechanicznych, lotniczych i energetycznych. Pakiet realizuje funkcje opisane w poprzednim punkcie (tym razem w odniesieniu do projektowania konstrukcji metalowych).

System nauczania metod ciągłej i dyskretnej optymalizacji deterministycznej — CADDOS

System CADDOS przeznaczony jest do komputerowego wspomagania nauczania wybranych metod optymalizacji deterministycznej. Algorytmy obliczeniowe systemu CADDOS zostały pogrupowane w ośmiu jednostkach tematycznych (ćwiczeniach) — programowanie liniowe, nieliniowe z ograniczeniami i bez ograniczeń; optymalizacja dynamiczna zero-jedynkowa, sieciowa; programowanie mieszane i dyskretne. W trakcie ćwiczeń studenci mają możliwość zapoznania się z dwoma podstawowymi — odpowiednimi dla każdego ćwiczenia — algorytmami oraz ugruntowania znajomości teorii optymalizacji.

Podsystem projektowania systemów zarządzania — DESOMA

Podsystem DESOMA, przeznaczony do wspomagania dydaktyki, związany jest z analizą i projektowaniem systemów zarządzania. Podsystem ten, zawierający 34 programy, służy do wyrabiania umiejętności i nawyków w posługiwaniu się informatyką, w analizowaniu procesów zachodzących w systemach zarządzania, a także w planowaniu i projektowaniu przedsięwzięć.

Podsystem DESOMA spełnia funkcje narzędzia umożliwiającego m.in.:

- modelowanie wybranych procesów
- estymację parametrów modeli wybranych procesów
- badanie sprzężeń zwrotnych w systemach gospodarczych
- wyciąganie wniosków dla celów prognozowania.

Każdy z programów podsystemu DESOMA stanowi autonomiczną całość i może być wykorzystywany niezależnie od innych.

Budowa atomu i cząsteczki

Pakiet programów z zakresu budowy atomu i cząsteczki służy do komputerowego wspomagania nauczania chemii, zwłaszcza w badaniach struktury i własności cząsteczki chemicznej. Pakiet ten umożliwia rozwiązywanie zagadnień związanych np. ze strukturą elektronową atomów i cząsteczek, obliczaniem wielkości fizykochemicznych itp. Pakiet zawiera osiem programów, które oprócz dydaktyki mogą być wykorzystane w pracach badawczych.

Metrologia przepływów w miernictwie energetycznym

Pakiet programów z zakresu metrologii przepływów w miernictwie energetycznym przeznaczony jest do obliczeń termoenergetycznych. Zastosowanie pakietu pozwala na wzrost dokładności obliczeń oraz przeprowadzenie szerszej analizy metrologicznej. Pakiet „Metrologia przepływów" składa się z następujących bloków tematycznych:

- metody pomiaru strumienia masy
- obliczanie podstawowych wielkości termodynamicznych nośników energii

• metody pomiaru strumienia masy przy badaniu maszyn i urządzeń energetycznych

• metodologia projektowania przepływomierzy zwężkowych o optymalnych własnościach metrologicznych.

System komputerowych symulatorów organizacji gospodarczych i gier kierowniczych — SYGRYDA

System SYGRYDA przeznaczony jest do kształcenia specjalistów w zakresie zarządzania, celem jego zastosowania jest ulepszenie procesu dydaktycznego poprzez umożliwienie prowadzącym zajęcia i słuchaczom wykonywania komputerowych eksperymentów symulacyjnych na gotowych symulatorach organizacji gospodarczych i prowadzenia gier kierowniczych, a także poprzez uruchamianie oraz testowanie własnych modeli i wykonywanie na nich eksperymentów.

System SYGRYDA składa się z następujących części, które mogą być używane niezależnie od pozostałych, a także niezależnie modyfikowane:

• Gra kierownicza „Inwestycje-Zapasy-Wynik Finansowy (IZWF), w której odtworzono system ekonomiczno-finansowy typowy dla przedsiębiorstwa działającego w gospodarce planowej o parametryczno-nakazowym systemie zarządzania.

• Symulator „Prognozy-Decyzje" (PDE1), będący zbiorem trzech modeli, które odwzorowują działanie hipotetycznego systemu produkcji i dystrybucji środków produkcji.

• Symulator „Zaopatrzenie" (ZAO1), w którym odwzorowano system zaopatrzenia organizacji przemysłowej typu kombinat, wraz z jego powiązaniami z dostawcą zewnętrznym i z własnym systemem produkcyjnym.

• Symulator „Remont" (REM1), w którym odwzorowano cztery strategie kierowania remontami oraz trzy strategie kierowania zamawianiem dostaw i własną produkcją części zamiennych.

• Język symulacji DYNAMIKA, służący do przygotowania komputerowych programów obliczeń symulacji ciągłej. Symulacja ta dotyczy zachowania się systemów, których model matematyczny tworzy zbiór równań różnicowych lub różniczkowych. Forma równań zapisywanych w języku DYNAMIKA jest identyczna z formą równań modelu matematycznego opisującego badany obiekt.

Systemy badań eksperymentalnych

Dla optymalnej organizacji badań eksperymentalnych przygotowano następujące rozwiązania informatyczne:

• komputerowy system eksperymentowania EXPES

• komputerowy system eksperymentowania na potrzeby dydaktyki EXPES-E

• pakiet programów z zakresu analizy wymiarowej.

System EXPES realizuje dwie podstawowe funkcje w procesie eksperymentowania: optymalną organizację eksperymentu oraz przetwarzanie wyników eksperymentu, zawiera także 11 programów dla ich realizacji.

System EXPES-E przeznaczony jest do dydaktyki z zakresu podstawowych i nowoczesnych metod i technik eksperymentowania wspomaganego komputerem. Umożliwia on prowadzenie zajęć dydaktycznych z metodologii eksperymentowania i identyfikacji obiektów sterowania. Zawiera 16 programów.

Pakiet programów analizy wymiarowej składa się z następujących grup programów:

— analizy wymiarowej

— przetwarzania wyników pomiarów tensometrycznych

— obróbki statystycznej procesów losowych (ukierunkowane na zastosowania w laboratoriach mechanicznych).

System TOK STUDIOW

Celem systemu informatycznego TOK STUDIÓW jest usprawnienie obsługi procesu dydaktycznego w szkole wyższej. System ten ewidencjonuje istotne informacje o każdym studencie od momentu jego immatrykulacji aż do uzyskania dyplomu ukończenia studiów. Baza informacyjna systemu umożliwia wydruk szeregu wydawnictw dla jednostek organizacyjnych zajmujących się obsługą procesu dydaktycznego, kierownictwa uczelni, resortu nauki, szkolnictwa wyższego i techniki oraz jednostek statystycznych. Emitując odpowiednie wyniki — usprawnia pracę sekretariatów wydziałów, odpowiedzialnych za przygotowanie i rozliczenie sesji egzaminacyjnej.

System selektywnej dystrybucji informacji (SDI/R)

Zadaniem systemu SDI/R jest regularne dostarczanie jego użytkownikom danych otrzymywanych od wyspecjalizowanych instytucji krajowych i zagranicznych. Zbiory danych zasilające system SDI/R w aktualne informacje zawierają opisy dokumentów (artykułów, patentów, książek itp.). Zainteresowania użytkowników, zarejestrowanych w systemie, reprezentowane są przez tzw. profile. W trakcie przetwarzania profile użytkowników porównywane są z opisami dokumentów ze zbiorów danych. W wyniku pracy sytemu użytkownicy otrzymują zestawienie literatury z interesujących ich dziedzin.

Zastosowany w systemie język formułowania profili pozwala na selekcję dokumentów według dowolnych cech (symboli klasyfikacji tematycznej, słów z tytułów publikacji, słów kluczowych, deskryptorów, tytułów czasopism — w których opublikowano dane prace, nazwisk autorów publikacji itp.). System nie nakłada ograniczeń na liczbę przetwarzanych zbiorów, liczbę obsługiwanych profili i użytkowników. Zastosowany w systemie moduł konwersji umożliwia przejście z dowolnego formatu zbiorów wejściowych na format akceptowany przez pozostałe moduły systemu.

* * *

Opisane powyżej systemy informatyczne i pakiety programów rozpowszechniane są przez Centrum MERA-ELWRO we Wrocławiu [1]), jako generalnego dostawcę systemów komputerowych. Centrum MERA-ELWRO proponuje użytkownikom następujące formy rozpowszechniania:

• sprzedaż

• dzierżawę, polegającą na odstąpieniu użytkownikowi systemu lub pakietu na czas określony (co najmniej rok) wg określonej zasady opłaty za dzierżawę i zabezpieczenia po zakończeniu okresu dzierżawy

• udostępnianie, polegające na przekazywaniu użytkownikowi wygenerowanych programów do realizacji konkretnego zastosowania.

W ramach dostaw systemów informatycznych dla szkół wyższych zapewnia się: dokumentację eksploatacyjną oraz bibliotekę programów binarnych.

Ponadto, w zależności od potrzeb odbiorcy, mogą być wykonane prace wdrożeniowe polegające na: przeszkoleniu personelu zajmującego się eksploatacją systemu lub pakietu programów oraz nadzorze autorskim w trakcie eksploatacji wstępnej i bieżącej. Dostawca zapewnia operatywne usuwanie błędów wykrytych w trakcie eksploatacji i dostarcza bezpłatnie ulepszone wersje oprogramowania.

ELŻBIETA KIERCZUK
BOGUSŁAW ZYGADŁO
CKSAiP MERA-ELWRO
Wrocław

[1]) Adres do korespondencji: Centrum MERA-ELWRO, Zakład systemów Użytkowych, ul. Ostrowskiego 30, 53-238 Wrocław, tel. 61-06-21 wew. 950.

GIEŁDA INFORMACJI

Publikujemy bezpłatnie:

- krótkie prezentacje nowych, nie wykorzystanych dostatecznie systemów
- doniesienia o innowacjach programowych
- ogłoszenia dotyczące nowych rozwiązań technicznych sprzętu informatycznego
- indywidualne oferty informatyków poszukujących pracy oraz informacje o wolnych etatach
- ogłoszenia w sprawie szkoleń informatycznych
- inne informacje ułatwiające doraźną organizację informatyki w Polsce.

SZKOŁA LETNIA LOGLANU

W dniach 5—10 września 1983 r. w dziewiętnastowiecznym pałacu w Zaborowie k.Warszawy odbędzie się Międzynarodowa Szkoła Letnia LOGLANU. Tematyka wykładów obejmie:

- składnię oraz formalne definicje semantyki języka (podejście operacyjne oraz aksjomatyczne)
- formalne specyfikacje struktur danych oraz ich LOGLANOWE realizacje
- współprogramy i procesy współbieżne
- metodologiczne podstawy języka
- zastosowania LOGLANU
- porównanie z innymi językami (ADA, SIMULA 67, MESA itd.).

Ponadto organizatorzy zapewniają możliwość praktycznego zapoznania się z kompilatorem LOGLANU. Szczegółowe informacje można uzyskać pod adresem: **Szkoła Letnia LOGLANU, Instytut Informatyki Uniwersytetu Warszawskiego, PKiN, VIII p., 00-901 Warszawa.**

SYMULATOR ODRA → RIAD

ZETO-Gdynia oferuje nowy typ wieloprogramowego dyskowego symulatora, który umożliwia wykonywanie programów ODRY s. 1300 na komputerach JS RIAD. Symulator ten zawiera wszystkie funkcje egzekutora E6RM, pracuje na komputerze JS RIAD bez systemu operacyjnego OS lub DOS. Jego obsługa jest identyczna z obsługą egzekutora ODRY s. 1300.

Liczba i wielkość jednocześnie wykonywanych programów jest ograniczona wielkością pamięci operacyjnej komputerów JS RIAD. Zbiory danych używane przez programy są przechowywane na odpowiednich nośnikach w formacie i kodzie ODRA. Umożliwia to natychmiastowe przenoszenie całych systemów informatycznych z komputerów ODRA na maszyny Jednolitego Systemu.

▶

SICOB'82

Nie zabrakło żadnej z wielkich firm komputerowych na zeszłorocznym, 33 salonie SICOB. Jak zwykle odbył się on w Paryżu, w dniach 22.09 — 1.10 i zakończył rekordową frekwencją 414 318 zwiedzających z całego świata.

SICOB'82 ujawni. tendencje panujące obecnie na rynku informatycznym. Mimo obfitości najnowocześniejszego sprzętu komputerowego (prezentowanego często przez samych konstruktorów), wbrew tradycji lat poprzednich, a podobnie jak w roku 1981, nie pokazano żadnej nowości w dziedzinie dużych maszyn uniwersalnych. SICOB należał do mini- i mikrokomputerów. Wiele uwagi poświęcono również terminalom i drukarkom. Dlatego też chcielibyśmy przedstawić Czytelnikom INFORMATYKI wybrane, ciekawsze eksponaty z tych czterech grup. Przytoczone parametry pozwolą lepiej zorientować się w obecnym poziomie techniki (np. nie jest już rzadkością w minikomputerze pamięć operacyjna o pojemności kilku M bajtów).

MINIKOMPUTERY

W dziedzinie minikomputerów gwiazdą wystawy był 32-bitowy PRIME 2250, całkowicie kompatybilny z resztą serii 50 (250 II, 550 II, 750 i 850) tej samej firmy. PRIME 2250 może pracować z 32 terminalami, jest wyposażony w pamięć o pojemności 512 K—4 M bajtów oraz mechanizm zarządzania pamięcią wirtualną do 512 M bajtów (adresowaną przez użytkownika). Model ten sprzedawany jest w czterech wersjach i przeznaczony głównie do zastosowań biurowych.

Firma PERKIN ELMER zaprezentowała mini 3250, który ma zastąpić model 3240 stając się od tej chwili komputerem największej mocy w swojej serii. Charakteryzuje się pamięcią 16 M bajtów oraz architekturą wieloszynową z szyną pamięciową (64 bity) i szyną multipleksera dla urządzeń peryferyjnych. Ta sama firma zapowiedziała nowy rodzaj pamięci dzielonej pomiędzy kilka (maksimum 16) procesorów serii 3000. Chodzi tu o system zorganizowany wokół szyny centralnej, z czterema blokami pamięci, każdy o pojemności 4 M bajtów.

Główny producent minikomputerów 32-bitowych, firma DIGITAL EQUIPMENT, wystawiła model VAX 11/730 prezentowany wcześniej w Hanowerze.

Firma MAI pokazała po raz pierwszy minikomputer MAI 810 z pamięcią operacyjną o 1,5—2 M bajtów, z możliwością przyłączenia 64 terminali.

Rozwój minikomputerów 22-bitowych nie oznacza bynajmniej śmierci maszyn 16-bitowych. Palma pierwszeństwa w dziedzinie nowych rozwiązań konstrukcyjnych minikomputerów 16-bitowych należy do firmy HEWLETT-PACKARD, która rozwija ostatnio swoje systemy serii HP 1000.

Nowością był też opracowany przez firmę PLESSEY PERIPHERAL SYSTEMS system 33 VHT uzupełniający gamę sprzętu pracującego na bazie LSI-11/23 (PDP-11/24 i PDP-11/34). System ten wyposażony jest w pamięć operacyjną 256 K bajtów (z możliwością powiększenia do 1 M) oraz jedną jednostką pamięci dyskowej typu Winchester. Użyto oczywiście listy instrukcji PDP-11/24.

NORSK DATA zaprezentowała tym razem nowy system ND-100/CX, ostatni ze swej serii minikomputerów 16-bitowych. Przeznaczony jest do zastąpienia ND-100/CE i dysponuje o 40% większą mocą obliczeniową.

Listę nowości uzupełniają IN 2027N firmy INTERTECHNIQUE oraz TANDEM 16 NON STOP i S/120 firmy DATA GENERAL.

Należy podkreślić, że w przeciwieństwie do ubiegłych lat, firma CII HONYWELL-BULL nie pokazała na tegorocznym SICOBIE żadnych istotnych nowości.

Natomiast relatywna cisza, spowodowana prawdopodobnie zaskoczeniem i nieufnością, towarzyszyła ekspozycji nowego minikomputera 48-bitowego! HARRIS 800 charakteryzuje się wybitnymi parametrami. Dysponuje pamięcią operacyjną 3 M bajtową oraz pamięcią wirtualną 12 M bajtów (administrowaną sprzętowo). Wspólne dla całej serii HARRIS oprogramowanie eksploatacyjne VULCAN pozwala na przyłączenie do 128 terminali pracujących jednocześnie w różnych trybach: interakcyjnym, z podziałem czasu, w czasie rzeczywistym itp. Ta różnorodność funkcji nie wpływa oczywiście na prędkość minikomputera.

MIKROKOMPUTERY

O nowych tendencjach mikroinformatyki świadczą niektóre prezentowane na wystawie modele komputerów osobistych (ang. *personal computers*).

OPUS — francuski mikrokomputer firmy SOBRELEC zbudowany na mikroprocesorze Z80, posiada 64 K bajty pamięci RAM, dwa interfejsy V24, modem ekran o przekątnej 24 cm i klawiaturę typu Azerty. Istnieje w dwóch wersjach: przenośnej (OPUS VS) i stacjonarnej (OPUS VT).

MICRAL 9050 — pierwszy własny mikrokomputer firmy CII-HB, zbudowany na mikroprocesorze 16-bitowym,

posiada 256 K bajtów pamięci operacyjnej (możliwość zwiększenia do 1 M), klawiaturę alfanumeryczną, ekran jednobarwny, cztery sprzęgi V24 i działa pod nadzorem sytemu operacyjnego QMOS. Jest oferowany w czterech wersjach.

AVC 777 — japoński mikrokomputer skonstruowany na mikroprocesorze Z80 A, jest wyposażony w pamięć o pojemności 64 K bajtów i drukarkę oraz pracuje pod nadzorem systemu operacyjnego CP/M 2.2

HP 75C — zaprezentowany przez firmę HEWLETT-PACKARD pojawi się w sprzedaży dopiero wiosną 1983. Posiada maksymalną pamięć 120 K bajtów (24 K bajty pamięci RAM, 48 K bajtów dla systemu operacyjnego i 3 moduły po 16 K bajtów pamięci ROM), klawiaturę typu Qwerty oraz ekran zbudowany na ciekłych kryształach. HP 75C może być używany jako sprzęt przenośny (25 cm × 12 cm × 3 cm) i jest wyposażony w jednostkę pamięci kasetowej HP 82161 oraz drukarkę znakową (szerokość wiersza 80 znaków). Do zastosowań biurowych przyłączany jest do drukarki znakowej typu HP 82905B, monitora ekranowego HP lub do odbiornika telewizyjnego. HEWLETT-PACKARD oferuje dla tego mikrokomputera oprogramowanie typu VISCALS, programy do opracowań graficznych, program transmisji danych oraz elektroniczny notatnik terminowy. Proponowana cena 995 dolarów USA.

DM 5 (Decision Mate 5) — prototyp mikrokomputera firmy NCR z pamięcią RAM o pojemności 64 K bajtów (z możliwością rozszerzenia do 256) oraz pamięciami na dyskach elastycznych o pojemności 320 K bajtów, NCR nie określił jeszcze momentu wprowadzenia tego modelu na rynek.

SG 2900 H — nowy mikrokomputer japońskiej firmy SHARP jest wyposażony w pamięć operacyjną o pojemności do 256 K bajtów oraz pamięć zewnętrzną o pojemności 450 M bajtów.

Równocześnie konstruktorzy japońscy zapowiedzieli kilka dalszych ulepszeń produktów już istniejących, jak np. dostępność systemu operacyjnego CP/M dla mikrokomputerów MZ 80 A i MZ 80 B.

Nowością jest także mikroprocesor MICRO/J-11, nie występujący na razie w żadnym konkretnym urządzeniu i stanowiący ofertę dla konstruktorów. Zbudowany w oparciu o technologię CMOS, jest on całkowicie dostosowany do wymagań systemów operacyjnych minikomputerów PDP-11, a także realizuje wszystkie funkcje minikomputera PDP-11/70. MICRO/J-11 akceptuje listę instrukcji komputerów PDP, realizuje sprzętowo operacje zmiennego przecinka oraz arytmetykę podwójnej precyzji. Oprogramowanie MICRO/J-11 może być przeniesione na minikomputery LSI-11, PDP-11 i VAX. Początek produkcji zapowiedziano na lato 1983.

DRUKARKI

Nowe rozwiązania drukarek prezentowane były już wiosną ub. r. w Hanowerze. Firmy japońskie HITACHI i RICOTT przedstawiały tam nowe modele drukarek laserowych. Na szeroką skalę stosuje się jednak w dalszym ciągu tradycyjne, dobrze opanowane technologie.

Największe zainteresowanie wzbudził w Paryżu prototyp drukarki magnetograficznej MP 6000 firmy CII-HB, mogącej śmiało konkurować z drukarkami laserowymi. MP 6000 drukuje z prędkością 88 stron na minutę, przy czym pojedyncza strona składana jest z elementów, które mogą być ułożone w stosunku do siebie pod pięcioma różnymi kątami, a sama strona może być drukowana pod kątem 90, 180 lub 270 stopni do kierunku ruchu papieru. Gęstość i rozmiar drukowanych znaków są zmienne (od 1 do 20 znaków/cal — w poziomie i od 1 do 18 wierszy/cal — w pionie). Drukarka ma rozmiary: 1,15 m × 1 m × 1 m.

W dziedzinie bardziej tradycyjnych rozwiązań firma CII-HB zaproponowała drukarkę CYNTHIA PR 54 o prędkości 1500 wierszy na minutę (z zestawem 48 znaków) lub 1200 wierszy/min (z zestawem 64 znaków).

CENTRONICS prezentował swoją pierwszą drukarkę wielobarwną 706 C pracującą w dwóch trybach: alfanumerycznym lub graficznym. Drukarka ta, wyposażona w matrycę 7 × 8 punktów, drukuje w czterech kolorach (czarnym, niebieskim, czerwonym i zielonym) i wyposażona jest w sprzęgi RS 232 C, CENTRONICS lub IEEE 488 do wyboru. Sprzedaż zapowiedziano na wiosnę 1983.

Nowe drukarki wielobarwne przedstawiły również firmy: TECHDATA (PRINCOLOR GP 1024) oraz PRIME (PRIM 8 i 132).

Technikę elektrofotograficzną wykorzystała firma AGFA-GEVAERT, w swej nowej drukarce P 400, która pracuje z prędkością 12 stron/min oraz posiada zmienny zestaw znaków pisarskich i symboli graficznych. Sprzęg RS 232 C umożliwia jej bezpośrednie przyłączenie do sieci telekomunikacyjnej.

Elektrofotografię stosuje również firma U-BIX dla nowego modelu drukarki PRINTER 1800. Drukarka ta pracuje z dwiema prędkościami (18 i 28 stron/min), umożliwia zwiększanie lub zmniejszanie rozmiarów liter i jest wyposażona w dwie kasety, które pozwalają na równoczesne wykorzystywanie papieru różnego formatu i koloru. Drukarka ma bufor pojemności 3 stron oraz trzy różne zestawy znaków (każdy po 96 znaków).

TERMINALE

33 SICOB obfitował w dużą różnorodność nowych terminali do zastoso-

GIEŁDA INFORMACJI

Do eksploatacji symulatora potrzebna jest odpowiednio szybka (od R-32 wzwyż) maszyna Jednolitego Systemu wyposażona w pamięć operacyjną o minimalnej wielkości 128 KB.

Koszt nabycia symulatora na każdą instalację wynosi 530 tys. zł. Przedmiotem sprzedaży jest: pakiet programowy (wynikowy), dokumentacja eksploatacyjna oraz usługi (instalacja pakietu, szkolenie). Dla odbiorców nabywających symulator na więcej niż jedną instalację bądź dla użytkowników poprzedniej wersji symulatora (jednoprogramowego i taśmowego) — stosowany jest opust.

Zamówienia prosimy kierować pod adresem: **ZETO-Gdynia, ul. Żwirki i Wigury 15, 81-968 Gdynia, tel. 21-66-46, teleks 054258, skr. poczt. 55.**

ZETOTEST

ZETO Łódź oferuje system ZETOTEST, pozwalający w stosunkowo prosty sposób generować zbiory taśmowe, niezbędne przy testowaniu i konserwacji oprogramowania użytkowego komputerów rodzin ODRA/ICL. Wymaga on w standardowej wersji 2—3 jednostek TM oraz po jednej — CK i DW, przy 32 KS pamięci operacyjnej. W jednym przebiegu maszynowym systemu ZETOTEST można wygenerować do 10 zbiorów taśmowych, wraz z czytelnie zredagowanymi tabulogramami kontrolnymi i syntetycznym raportem przebiegu testowania.

Praktyka wykazała, że system ten zapewnia: wysoką jakość zbiorów i wyeliminowanie wpływu subiektywnego punktu widzenia poszczególnych wykonawców oprogramowania użytkowego, istotne usprawnienie procesu odbioru oprogramowania od wykonawców oraz elastyczność harmonogramowania prac w zespole projektowym. ZETOTEST umożliwia badanie działania programu w przypadkach nietypowych i stanach krytycznych. Pozwala w znacznym stopniu zmniejszyć pracochłonność.

Kontakt merytoryczny: mgr inż. Zdzisław K. Dąbrowski, Oział Oprogramowania Narzędziowego, tel. 829-13 w. 41; Informacje ogólne: Dział Handlowy, tel. 829-13, w. 72; Adres korespondencyjny: ZETO-Łódź, ul. Narutowicza 136, 90-146 Łódź. Teleks 884314 ZETO PL.

TEZAURUS WIELODZIEDZINOWY

W Instytucie Technologii Materiałów Elektronicznych Centrum Naukowo-Produkcyjnego Materiałów Elektronicznych w Warszawie został opracowany „Tezaurus materiałów elektronicznych o zakresie obsługi informacyjnej Instytutu Technologii Materiałów Elektronicznych CNPME, do celów informacji naukowej". Tezaurus ten zawiera 3731 terminów kluczowych, w tym: 3538 deskryp-

▶

GIEŁDA INFORMACJI

torów, opisujących tematykę prac naukowo-badawczych ITME i 193 askryptory, mające swoje znaczeniowe odpowiedniki w reskryptorach.

Tezaurus składa się z:
— alfabetycznego zestawu zbiorczego par terminów
— alfabetycznego wykazu zależności hierarchicznych deskryptorów
— alfabetycznego wykazu askryptorów
— spisu literatury, z której dokonano ekscerpcji terminów kluczowych.

Trudności budowy wspólnych baz danych (obejmujących wiele dziedzin nauki i techniki) metodą integracyjną skłoniły Instytut do opublikowania opracowanego tezaurusa, jest on bowiem przykładem tezaurusa wielodziedzinowego, pomocnym dla instytucji opracowujących własne tezaurusy dziedzinowe.

Tezaurus zostanie wydrukowany w ramach małej poligrafii (objętość ok. 260 stron). Zainteresowani jego nabyciem powinni przesłać pisemne zamówienie pod adresem: **Instytut Technologii Materiałów Elektronicznych ZOINTE, 02-673 Warszawa, ul. Konstruktorska 6, tel. 43-74-61 (do 5) lub 43-54-24 (do 8) wew. 123.**

DLA KIEROWNIKÓW

Centrum Szkolenia Informatycznego CSI przy ZETO-Łódź organizuje w terminie 24—26 maja 1983 seminarium dla kadry kierowniczej ośrodków obliczeniowych. Tematyka seminarium obejmie:
— organizację pracy i stosowane technologie
— dokumentowanie systemów informatycznych
— newralgiczne elementy zarządzania ośrodkiem obliczeniowym
—aktualny stan i tendencje rozwoju informatyki we Francji.

Na adres CSI należy przesyłać ok. 2—3 stronicowe streszczenia referatów, które zostałyby wygłoszone na seminarium. Zamknięcie listy zgłoszeń nastąpi 31 marca br. Wszyscy prelegenci zapraszani są na koszt organizatora.

Warunkiem uczestnictwa innych osób w seminarium jest przesłanie na adres: **CSI ZETO-Łódź, ul. Hutora 69, 91-558 Łódź** — imiennego zgłoszenia zawierającego oświadczenie o wpłaceniu na konto: ZETO-Łódź, NBP I/Ó Łódź nr 47018-2219, kwoty 4000 zł od osoby. Wszyscy uczestnicy zostaną zakwaterowani w Ośrodku Szkoleniowo-Wypoczynkowym ZETO-Jelenia Góra (przy ul. Myśliwskiej 14) w Jagniątkowie k.Jeleniej Góry, gdzie będą odbywały się obrady seminarium.

Dodatkowe informacje w CSI: tel. 647-70, teleks 885208 ZETO PL.

wań specjalnych. Zaprezentowano również całą gamę małych, lekkich terminali kieszonkowych.

Firma MSI wystawiła najmniejszy z serii — model 55 (13,5 cm × 6 cm × × 1,9 cm, 180 g) oraz najnowszy terminal programowalny — MSI 85.

MATRA pokazała terminale ASCII TEE 415 i 720 służące do komunikowania się z bazami danych poprzez linie telefoniczne komutowane lub dzierżawione.

Nie zabrakło również nowych terminali na stoiskach czołowych producentów komputerów: IBM prezentował terminal IBM 5292 (kolorowy lub jednobarwny) przeznaczony dla komputerów IBM 34 lub 38. SIEMENS wzbogacił swoją ofertę modelami 9751 (terminal uniwersalny), 9752 (terminal wielobarwny) oraz 9753 (terminal inteligentny zbudowany na mikroprocesorze INTEL 8085A, wyposażony w pamięć o pojemności 64 K bajtów). INTERTECHNIQUE wprowadził model 4 M 407, a ERICSSON DATA SYSTEMS — model ALPHASKOP S 41.

TEKTRONIX rozszerzył możliwości dotychczasowego modelu terminala 4112, uzupełniając go zbiorem 4096 kolorów. Nowy model TEK 4113 pozwala jednocześnie operować wybranymi przez siebie 8 kolorami, ma ekran panoramiczny i umożliwia lokalne programowanie, które znacznie odciąża główny komputer.

* * *

Zgodnie z wieloletnią tradycją, otwarcie salonu SICOB'82 poprzedziła międzynarodowa konferencja CONVENTION INFORMATIQUE.

Tradycyjne sprzężenie obu imprez zostanie przerwane w bieżącym roku, wyjątkowo w dniach od 30 maja do 1 czerwca 1983 r. odbędzie się konferencja wiosenna Printemps Convention). Natomiast SICOB będzie towarzyszyć jesienią światowemu kongresowi IFIP — najbardziej prestiżowej konferencji w dziedzinie informatyki. Można się zatem spodziewać, że za rok producenci pokażą jeszcze więcej atrakcyjnych nowości.

Oprac. MARIANNA SOBCZYK na podstawie „01 informatique hebdo", nr 715—718

CONVENTION INFORMATIQUE

Podobnie jak SICOB, CONVENTION INFORMATIQUE osiągnęła najwyższą frekwencję — 3312 uczestników z 43 krajów (w roku 1981 — 3175 uczestników z 40 krajów). Pomimo pewnej stagnacji gospodarczej, wiele dziedzin informatyki nadal dynamicznie się rozwija. Wynika to zarówno z danych dotyczących liczby i wartości sprzedawanego i instalowanego sprzętu komputerowego, wartości sprzedawanych usług informatycznych, jak i zainteresowania informatyką nie tylko profesjonalistów (co zrozumiałe), ale i szerokiej publiczności.

Informatyka stała się jednym z czołowych tematów prasy codziennej. W okresie trwania SICOBU i C.I., „Le Monde" zamieścił szereg ważkich artykułów, które następnie wydał w specjalnym ilustrowanym dodatku. Nagłówki artykułów z pierwszej strony oddają w dużym stopniu sytuację informatyki francuskiej: „Ostatnia szansa dla Francji", „Dziesięć lat informatyki europejskiej czyli stracone okazje", „Podjąć środki przeciwko wyzwaniu elektroniki". Oczywiście, mowa tu o sytuacji Francji w stosunku do informatycznej czołówki świata — Stanów Zjednoczonych i Japonii.

„Le Monde" wydał też specjalny zbiór artykułów pt. „Informatyka dnia dzisiejszego" — 150 stron dużego formatu, ozdobnym drukiem — w sprzedaży we wszystkich kioskach. Informatyka stała się już czynnikiem życia codziennego. W dużym salonie „FNAC", sprzedającym aparaturę radiową, telewizyjną i fotograficzną, można zobaczyć młodych ludzi zabawiających się mikrokomputerami, które służą tu do zabawy, do nauki języka czy nauki programowania w BASICU. Informatyka wychodzi już niemal na ulice.

CONVENTION INFORMATIQUE odbywała się w paryskim Pałacu Kongresów, nowoczesnym „kombinacie" obsługującym różnego rodzaju kongresy i sympozja, w dn. 20—24 września ub.r., pod ogólnym hasłem: „Rok 1982 — Nowa Informatyka". Program konferencji składał się z czterech głównych grup tematycznych: technika, aspekty ekonomiczne informatyki, aspekty socjalne i prawne, zastosowania w różnych dziedzinach.

Omówienie prawie 300 referatów zawartych w dwóch tomach o objętości 800 stron dużego formatu przekracza

ramy i możliwości niniejszego artykułu. Ograniczę się więc tylko do kilku ogólnych uwag.

Nowe technologie przechowywania informacji. Przewiduje się, że w niedalekiej przyszłości systemy informatyczne będą wyposażone w dyski optyczne, numeryczne, na których za pomocą lasera będzie można zapisać 10 mld bitów. Na dwóch stronach dysku o średnicy 30 cm będzie można zapisać ponad milion stron maszynopisu.

Nowe technologie wejścia i wyjścia informacji. Firma IBM pracuje obecnie nad skonstruowaniem maszyny do pisania, w której tekst będzie wprowadzany głosem. Proszę sobie wyobrazić taki obrazek — pracownik dyktuje do mikrofonu, a w niedalekim pokoju maszyna pisze s a m a dyktowany tekst, lub ewentualnie — tekst zapisany wcześniej na taśmie czy dysku magnetycznym.

Nowe metody opracowywania oprogramowania. W chwili obecnej sprzedawane są specjalizowane minikomputery z oprogramowaniem pomagającym w opracowaniu zarówno samych programów, jak i dokumentacji technicznej oraz analizy funkcjonalnej systemu.

Mikroprocesory i mikrokomputery — najbardziej dynamiczna dziedzina informatyki — roczne przyrosty mierzone wartością sprzedaży sięgają 40 i więcej procent. Mikrokomputery wykorzystywane są w wielu nowych dziedzinach. Warto wspomnieć, że szereg firm samochodowych stosuje już mikrokomputery do wieloaspektowej informacji o pracy silnika. Szeroko stosowany jest też tzw. *telepayment;* specjalną plastikową kartę z wmontowanym wewnątrz mikroprocesorem wkłada się w sklepie do terminala przyłączonego do komputera w banku. Na koncie jest zarejestrowana suma pieniędzy, jaką dysponuje posiadacz karty. Suma do zapłacenia odejmowana jest przez mikroprocesor od sumy pierwotnie dysponowanej przez posiadacza karty, a terminal przekazuje realizowaną transakcję do banku na konto sklepu.

Aspekty ekonomiczne informatyki — sesje im poświęcone cieszyły się szczególnym powodzeniem. Wiele przedsiębiorstw w informatyce szuka środka na obniżenie kosztów, co w dobie trudności gospodarczych i zwiększającej się stale konkurencji ma istotne znaczenie. Jeszcze niedawno informatyka była drogą zabawką, którą kupowało się w momencie dobrej koniunktury przedsiębiorstwa (gdy było dużo pieniędzy i można było sobie pozwolić na dodatkowe inwestycje), dziś nahomiast kupuje się i instaluje komputery, aby uchronić się od przegranej w walce z konkurencją. Przedsiębiorstwa często nie mogą sobie pozwolić na niestosowanie informatyki. Użytkownicy komputerów natomiast starają się uzyskać jak najwięcej informacji dotyczących kosztów, a także możliwych efektów informatyzacji.

Aspekty socjalne i prawne informatyki. Poruszano zagadnienia związane ze zmianami w organizacii i warunkach pracy, jakie wprowadziło szerokie stosowanie informatyki; w tym — zagadnienia zmiany miejsca pracy. Rozważano możliwość wykonywania pracy w domu, dzięki np. zainstalowaniu domowego terminala. Wprowadzenie tej formy pracy radykalnie zmienia stosunki międzyludzkie związane z pracą zawodową (niedawno opublikowano wywiad z pracownicą banku, u której w domu zainstalowano terminal). Omawiano też m.in. wpływ informatyki na zmianę sytemu nauczania.

Zagadnienia formalno-prawne obejmowały głównie kwestie finansowe. Warto na marginesie wspomnieć, że Francja jest pierwszym krajem, który wprowadził u siebie przepisy prawne dotyczące ochrony własności informacji oraz ochrony obywateli przed nadużywaniem ich informacji osobistych.

Ostatnią grupą zagadnień poruszanych na C.I. były zastosowania w poszczególnych dziedzinach gospodarki, m.in. w przemyśle rolno-spożywczym, bankowości, handlu. Osobne podgrupy stanowiły doświadczenia w dziedzinie banków danych, telematyki (teleinformatyki), biurotyki, sterowania produkcją oraz zastosowań inżynierskich.

B i u r o t y k a (fr. *bureautique*, ang. *office automation*) — zastosowanie informatyki w pracach biurowych w celu przetwarzania sformalizowanych poleceń i tekstów w sposób zautomatyzowany. Biurotyka swym zasięgiem obejmuje zespół urządzeń do wprowadzania, zapamiętywania, przetwarzania, przesyłania i rozpowszechniania informacji. Przekładając to na język potoczny — biurotyka obejmuje zespół środków i technik służących do automatyzacji prac biurowych, a w szczególności do przetwarzania i przekazywania informacji w postaci słownej, pisanej lub graficznej.

T e l e m a t y k a (teleinformatyka) — fr. *télématique*, ang. *telematics* — neologizm wprowadzony w 1978 roku dla określenia zespołu technik i usług, które stosuje jednocześnie informatyka i telekomunikacja.

Biurotyka charakteryzuje się ogromnym rozwojem i możliwościami. Dla zobrazowania wagi problemu przytoczmy kilka cyfr: w USA w latach 1960—1980 produkcja przemysłowa wzrosła 13 razy, liczba robotników tylko 2 razy, natomiast liczba pracowników biurowych — 12 razy; we Francji przewiduje się, że w latach 1980—1985 na 5 nowotworzonych miejsc pracy aż 4 będą w biurach, a obecnie już prawie połowa ludności aktywnej zawodowo (ok. 12 mln) zajmuje się pracą biurową. Nic też dziwnego, że (szczególnie w USA) od dłuższego już czasu intensywnie inwestuje się w tej dziedzinie.

Jedną z podstawowych dziedzin biurotyki jest obecnie automatyczne przetwarzanie tekstów. Do chwili obecnej podstawowym rozwiązaniem było tu urządzenie jednostanowiskowe, stanowiące połączenie elektrycznej maszyny do pisania, niewielkiego procesora i pamięci na dyskach elastycznych. Obecnie coraz częściej stosuje się urządzenia wielostanowiskowe z dużą wspólną pamięcią, połączone w ramach przedsiębiorstwa wewnętrzną siecią transmisji danych. Następnym etapem rozwoju jest połączenie maszyny do przetwarzania tekstów z teleksem. W ten sposób dochodzimy do pojęcia „elektrycznej poczty" *(curier electronique)*. Po prostu, zamiast po automatycznym przetworzeniu tekstu drukować list, zostaje on przesłany za pomocą linii telefonicznej do odbiorcy. U odbiorcy może on być wydrukowany lub pokazany na ekranie i zapamiętany w pamięci sytemu.

Głośnym we Francji rozwiązaniem stała się „komputerowa książka telefoniczna". W jednym z departamentów zamiast stale drukować wymagającą częstej aktualizacji książkę telefoniczną, zaopatrzono wszytkich abonentów w bardzo proste, tanie terminale z małym ekranem. Za pomocą tego terminala można uzyskać wszytkie zarejestrowane w pamięci komputera informacje, jakie dawniej znajdowały się w książce telefonicznej. Terminal taki jest dzierżawiony przez urząd poczt i telefonów za niewielką opłatą miesięczną 70 franków. Obecnie całe sztaby specjalistów zastanawiają się, jakie jeszcze inne usługi można będzie świadczyć za pomocą tego terminala, oczywiście — za dodatkową opłatą.

Chciałbym jeszcze zwrócić uwagę Czytelnikom na sytuację kadrową w informatyce.

Olbrzymia ilość nowych zastosowań związanych z coraz większymi możliwościami oraz większą liczbą i różnorodnością maszyn powoduje lawinowy wzrost zapotrzebowania na nowe sytemy i programy. Pomimo znacznego rozwoju oprogramowania, w tym systemów wielokrotnego stosowania, oraz znacznego postępu w metodach projektowania i programowania — rynek informatyczny cierpi na poważny brak fachowców-informatyków. Liczba ofert dla różnych specjalności informatycznych wzrasta, np. we wrześniu ub.r. osiągnęła we Francji rekordowy stan 2342, w sytuacji pewnej stagnacji w innych dziedzinach gospodarki. Ponieważ tendencja ta istnieje od dłuższego czasu, wpływa to w istotny sposób na poziom zarobków w tej grupie zawodowej, który jest wyższy niż w innych porównywalnych zawodach.

Ponadto aktualny rozwój zastosowań oraz wzrost kosztów osobowych spowodował bardzo istotne zmiany w kosztach informatyzacji przedsiębiorstwa. Obecnie koszty związane z opracowaniem, konserwacją i modernizacją oprogramowania przekraczają niekiedy 50% łącznych wydatków na komputeryzację przedsiębiorstwa. W konsekwencji ogromnego rozszerzenia rynku systemów informatycznych nastąpił dynamiczny rozwój przedsiębiorstw usługowych w tej dziedzinie. Przedsiębiorstwa te osiągają roczne przyrosty obrotów rzędu 20—40%, a największe z nich, zatrudniające po 2000—3000 pracowników miały osiągnąć w ubiegłym roku obroty rzędu 1 mld franków. Nastąpiła istotna „profesjonalizacja" zawodu informatyka — staje się on pracownikiem dużego przedsiębiorstwa usługowego.

Na razie, zarówno przedsiębiorstwa usługowe, jak i informatycy — optymistycznie patrzą w przyszłość. Wszystkie prognozy wskazują na dalszy rozwój informatyki, która stała się niekwestionowanym już przez nikogo nośnikiem postępu gospodarczego.

MICHAŁ ZIĘBIŃSKI

RECENZJE

Próba czasu

Polscy czytelnicy otrzymali tłumaczenie jednej z najbardziej wartościowych książek o mikroprocesorach, napisanych do 1977 roku [1]).

Zamiarem autora było wszechstronne przedstawienie zagadnień związanych z budową systemów mikroprocesorowych, od zasad działania układów cyfrowych o małym i średnim stopniu scalenia, do syntezy systemów. Nawet częściowa realizacja tak ambitnych założeń może stanowić o wartości książki.

Jej treść można podzielić na trzy wzajemnie powiązane części. W pierwszej części (rozdziały 1—4) dokonano krótkiego przeglądu zagadnień projektowania, omówiono podstawowe zasady działania układów stanowiących elementy konstrukcyjne urządzeń cyfrowych (urządzenia pamięciowe, układy kodujące i inne bloki o średnim stopniu scalenia) oraz przedstawiono typowe przykłady takich elementów.

Część druga (rozdziały 5—8) jest poświęcona wprowadzeniu podstawowych pojęć z dziedziny techniki komputerowej (procesor i jego lista rozkazów, pamięć, podsystem wejścia-wyjścia) oraz omówieniu metody projektowania mikrokomputera opartego na mikroprocesorze INTEL 8008.

W ostatniej części (rozdziały 9—14) omówiono bardziej szczegółowo zagadnienia systemów mikroprocesorowych, tzn. języki i tłumaczenie programów, struktury adresowe, struktury i techniki operacji zewnętrznych, struktury przerwań, bezpośredni dostęp do pamięci oraz zagadnienia architektury.

Konieczność tak szczegółowego omówienia niektórych zagadnień autor uzasadnia dość przekonywująco (str. 169): *„Poeta, który nie zna ortografii i nie jest pewien znaczenia swoich słów, raczej nie będzie twórcą wielkiej poezji. Projektant, który nie przyswoi sobie szczegółów działania dotyczących każdego bloku systemowego i ma tylko mgliste pojęcie o ich działaniu — nie wyjdzie poza przeciętność".* Niestety, książka musi mieć ograniczone rozmiary, dlatego ilość miejsca, jakie można było poświęcić omówieniu zarówno podstawowych układów TTL, jak i układów LSI, wystarcza jedynie do tego, aby czytelnik uzyskał *„mgliste pojęcie o ich działaniu".*

Próba przedstawienia systemów, ilustrowana opisem kilkunastu procesorów, powiodła się dzięki doskonałej prezentacji zagadnień związanych z architekturą systemów mikroprocesorowych. Niektóre rozdziały, np. dotyczące operacji zewnętrznych, przerwań, bezpośredniego dostępu do pamięci — napisane są wyjątkowo przystępnie. Rozdziały, w których autor wyraźnie obniża loty (np. — dotyczące podsystemów pamięci i wejścia-wyjścia) można bez większej szkody pominąć, gdyż informacje w nich zawarte nie mają obecnie większego znaczenia praktycznego.

W książce opisano wiele mikroprocesorowych układów o dużym stopniu scalenia. Mimo że autor używa w przykładach historycznego już procesora INTEL 8008, wiele miejsca poświęco także innym układom, jak: procesor INTEL 8080, pamięć RAM typu 2102, układ transmisji szeregowej typu 8251, a nawet mikroprocesor INTEL 8048.

W książce jedynie zarysowano tendencje rozwojowe, tj. pojawienie się programowanych układów wspomagających, struktur wieloprocesorowych i procesorów o większej długości słowa. Przedstawienie zagadnień programowania systemów mikroprocesorowych ograniczono do omówienia makroasemblera. Nie możemy jednak zapominać, że dopiero w 1978 r. pojawił się jeden z najbardziej popularnych obecnie mikroprocesorowych systemów operacyjnych, tzw. CP/M. Nie istniały również w tym czasie kompilatory języków wysokiego poziomu, poza dalekimi od doskonałości pierwszymi wersjami kompilatorów języka PL/M.

Oczywiście, pięć lat rozwoju mikroprocesorów to bardzo dużo. Wpływ czasu spowodował nieadekwatność tytułu i treści książki. Fragmenty poświęcone właściwemu projektowaniu są mniej istotne. Nie sądzę jednak, aby ktokolwiek przewidywał, że w ciągu niewielu lat mikroprocesory istotnie zagrożą współczesnym dużym komputerom.

Choć można mieć wątpliwości, czy celowe jest tłumaczenie książek z dziedziny, której szybkość rozwoju jest niespotykana w technice, moim zdaniem książka Klingmana w ogólnym przekroju wychodzi z próby czasu zwycięsko. Komu można tę książkę polecić? Niewątpliwie, może być ona użyteczna dla studentów elektroniki, informatyki, automatyki i kierunków pokrewnych, którzy specjalizują się w zagadnieniach systemów mikroprocesorowych. Jednak specjaliści z innych dyscyplin, nie tylko technicznych, dzięki lekturze tej książki mogą uzyskać szerszy pogląd na dziedzinę techniki, która będzie mieć trudny do przewidzenia wpływ na życie ludzi i społeczeństw.

Tłumaczenie książki nie budzi zastrzeżeń, zarówno językowych, jak i terminologicznych. W porównaniu z tłumaczeniami innych książek z dziedziny elektroniki lub informatyki, można je uważać za wzorowe. Tłumaczom udało się przekazać pasję autora, z którą stara się on przekonać czytelników o znaczeniu i swoistym pięknie systemów cyfrowych.

Do wydawcy można mieć pretensję o brak, chyba koniecznej, przedmowy do wydania polskiego i — o niewykorzystanie możliwości poprawienia oryginału przez umieszczenie podpisów pod licznymi i bardzo dobrymi rysunkami.

LEONID BUŁHAK

[1]) Edwin E. Klingman: Projektowanie systemów mikroprocesorowych. WNT, Warszawa, 1982. Tłum. z angielskiego: A. Michalski, M. Thor, J. Zalewski. Dane oryginału: Microprocessor Systems Design, Prentice-Hall, Englewood Cliffs, NJ, 1977

Trzy słowniki

Terminologia informatyczna z trudem poddaje się przemianom. Zwłaszcza w przypadku naszego języka. Wychodzi to zwłaszcza na jaw przy tłumaczeniu tekstów źródłowych dotyczących nowego tematu, gdyż wówczas powstają najczęściej prowizoryczne określenia zastępcze, w dodatku wielowyrazowe, które mimo różnych niedogodności bardzo trudno potem wyplenić. Z drugiej jednak strony, gdyby „słownikarze" czekali aż dana dziedzina dorobi się bezdyskusyjnej terminologii, to właściwie niczego nie mogliby wydać — może poza dziedzinami „martwymi", wokół których dawno wygasły emocje.

Strategia porządkowania terminologii informatycznej ma do dyspozycji dwie metody, które nie gwarantują — co prawda — generalnych sukcesów, ale długo zachowują skuteczność oddziaływania. Umownie można te metody nazwać b e l f e r s k ą i b i b l i o f i l s k ą. Pierwsza polega na ograniczaniu się do przypadków bezdyskusyjnych, druga zaś — na odwrót — z lubością wyżywa się w zestawieniu drobiazgowych list terminologicznych.

Słowniki pierwszego rodzaju, które można określać mianem k i e s z o n k o w y c h, mają zazwyczaj bogaty rodowód. Taki słownik można najrzetelniej sporządzić jako wyciąg z większego dzieła. W naszym kraju ta sztuka nikomu się jeszcze nie udała. Ale oto na Międzynarodowych Targach Książki w Warszawie wpadła mi w ręce interesująca pozycja. Jest to *Kieszonkowy leksykon informatyki*[1]) trójki autorskiej Löbel-Müller-Schmid, której większy słownik znany jest u nas w przekładzie Władysława Klepacza (por. INFORMATYKA 5/79). Właśnie z tego większego słownika wybrano 750 najważniejszych haseł, których teksty maksymalnie skrócono, a wszystkie ilustracje i tabele przesunięto do osobnego aneksu. Tylko kilka haseł ma dłuższe — do półtorej szpalty — omówienia.

Oczywiście, dla fachowców takie wydawnictwo jest zbędne. Ale przecież istnieją tysięczne rzesze wykładowców, nauczycieli, studentów, instruktorów, kursantów, istnieje poszerzający się krąg użytkowników będących laikami w dziedzinie informatyki — którym potrzebne są standardowe definicje. Toteż brak kieszonkowca typu *EDV Taschenlexikon* w polskich księgarniach jest dotkliwym mankamentem.

W związku z drugim typem słowników wśród informatyków krążą legendy. Choćby o „kartotece rodziny Sipplów", ponoć założonej w postaci komputerowej bazy danych i systematycznie rozszerzanej do ponad 30 tys. terminów. Rekord światowy ma zaś należeć do komputerowego katalogu cytatów terminologicznych, jaki od lat prowadzi koncern Siemensa, a którego objętość można oszacować na kilkaset tysięcy pozycji.

Sipplowie co kilka lat wydają specyficznie zdefiniowane podzbiory swej stale rozszerzanej kartoteki. W ten sposób wydali oni od połowy lat sześćdziesiątych już kilkanaście różnych słowników. Nigdy jednak nie próbowali wydać całej kartoteki — swego rodzaju maksymalnego słownika. Byłoby to dzieło kosztowne, w dodatku nieporęczne ze względu na swą wielkość; nie mogłoby liczyć na masowy popyt. A i tak co kilka lat wymagałoby drukowania suplementów.

Targowi bywalcy mogli na stoisku *Prentice-Hall* w warszawskim PKiN przejrzeć aż dwa słowniki sipplowskie. Słowniki te wiążą się z działalnością przedsiębiorstwa Howarda W. Samsa i Spółki z Indianapolis, wyspecjalizowanego w organizowaniu różnych kursów szkoleniowych, na potrzeby których wydaje we własnym zakresie odpowiednie zeszyty do ćwiczeń, materiały metodyczne oraz słowniki terminologiczne i teksty encyklopedyczne. Mogłoby to być wskazówką dla naszych czołowych ośrodków szkoleniowych, dla których porządny słownik byłby dopiero podstawą skutecznej pracy.

Słownik Komputerowy[2]) oraz *Słownik Mikrokomputerowy*[3]) w porównaniu do poprzednich wydań różnią się przede wszystkim odmiennym układem dodatków encyklopedyczno-podręcznikowych. Znikły opisy języków QUICK-TRAN i CAL — na korzyść RATFOR, APL, MUMPS, CONTROL BASIC. Powstały też nowe skróty.

W II wydaniu *Słownika Komputerowego* wyodrębniono osobne listy skrótów sprzętowych, programowych i telekomunikacyjnych — które w obecnym wydaniu zostały scalone w jedną listę i rozbudowane. Jednocześnie jednak szereg haseł uległo przebudowie albo wręcz usunięciu. Słownik ten ma charakter podręcznego narzędzia dla osób praktycznie niezainteresowanych całym balastem historycznym komputeryzacji. Zawiera 12 tys. haseł.

Nieco mniejszy jest *Słownik Mikrokomputerowy*, obejmujący tylko ok. 9 tys. haseł, w tym ponad 5 tys. podstawowych. Jako swego rodzaju materiał pomocniczy, słownik ten również nie zawiera żadnych wskazówek bibliograficznych — i w tym sensie stanowi świadome przeciwieństwo omawianej swego czasu na naszych łamach (INFORMATYKA 8/77) encyklopedii Ralstona-Meeka, pochodzącej zresztą od innego wydawcy. Już przy pobieżnym kartkowaniu uderza w oczy duża liczba rysunków i fotografii umieszczanych bezpośrednio przy definicjach. Równie dużo materiału ilustracyjnego — ale już wyłącznie fotograficznego — zamieszczono w dwu aneksach encyklopedycznych, opisujących systemy mikrokomputerowe, minikomputerowe i ich mikroprocesorowe „cegiełki". Porównując obecne wydanie z pierwowzorem sprzed przeszło siedmiu laty można doszukać się ok. 1200 nowych bądź rozszerzonych haseł.

I tylko przykro jest, że czytelnik krajowy — aby móc zajrzeć do nowo wydawanych za granicą słowników informatycznych — w praktyce jest skazany wyłącznie na imprezy targowe. Przybliżenie takich dzieł informatykom jest niezbędne. A swoją drogą, można się zastanowić nad niewątpliwym faktem, że słowniki informatyczne powstają na Zachodzie w zasadzie poza nurtem głównych wydawców książek informatycznych, jakby to była działalność komercjalnie niezbyt pociągająca. Sukcesy wydawnicze firmy monachijskiej oraz indianopolisańskiej zadają kłam takim obawom, ale wyłącznie dzięki odpowiedniej wieloletniej strategii wydawniczej.

ADAM B. EMPACHER

---

[1]) Guido LOBEL, Peter MÜLLER, Hans SCHMID: EDV Taschenlexikon; Verlag Moderne Industrie, IX wyd., Monachium 1981; 230 str.; 28 rys. + 14 tabl.

[2]) Charles J. SIPPL, Roger J. SIPPL: Computer Dictionary and Handbook. Howard W. Sams & CO (wydawca), Indianapolis 1980; Prentice-Hall International (dystrybutor). D o d a t k i: wprowadzenie do informatyki (komputery osobiste, rynek komputerowy, systemy komputerowe, cybernetyka i sztuczna inteligencja, zastosowania), bazy danych (systemy zarządzania bazami danych, aspekty techniczne), teoria zarządzania przedsiębiorstwami (metody, techniki, rola komputerów, szkolenia kadry), kryteria systemowe (pułapki komputeryzacji, proces projektowy, analiza efektywności), modelowanie systemów (analiza systemowa, klasy modeli, etapy badawcze), badania operacyjne (organizacja prac, szkolenie kadry, metody realizacyjne), słowniczek matematyczny (600 definicji), słowniczek statystyczny (350 definicji), układy numeracji (dwójkowy, ósemkowy, szesnastkowy, kody, tabliczki logiczne), schematy blokowe (systemowe i programowe, oznaczenia), języki komputerowe (klasyfikacja i przykłady), przetwarzanie tekstów słowniczek skrótów, problemy przyszłości (inwazja mikroprocesorów, wielkie sieci komputerowe, zastosowania heurystyczne).

[3]) Charles J. SIPPL: Microcomputer Dictionary. Howard W. Sams & Co. (wydawca), Indianapolis 1981; Prentice-Hall (dystrybutor). D o d a t k i: mikroprocesory (podłoża półprzewodnikowe, układy 8-, 16- i 32-bitowe, VLSI, CMOS, EPROM, sprzęgi, pamięci pęcherzykowe, syntezatory mowy, wideodyski), mikrokomputery (rynek, zastosowania, przegląd typów walizkowych i kieszonkowych, roboty a mikrokomputery, sieci VIEWDATA i TELETEXT).

Słowniczek mikroprocesorowy (cz. 1)

Specyfikę mikroprocesorów określa przede wszystkim ich architektura, a także niektóre właściwości i parametry (łącznie z rozmiarami, ceną itp.). Z drugiej jednak strony — są to po prostu miniaturowe procesory (komputery) i z tego względu redakcja INFORMATYKI — doceniając potrzebę wydania *Słowniczka mikroprocesorowego* — uznała zarazem, że czytelnicy nie oczekują definiowania haseł od dawna znanych i stosowanych w technice komputerowej, jak np. pamięć, program, przerwanie. Wskutek tego liczba haseł została maksymalnie ograniczona, a w ich doborze — z natury rzeczy zawsze subiektywnym — preferowano hasła bądź zupełnie nowe, bądź szczególnie charakterystyczne dla mikroprocesorów.

Do najtrudniejszych — jak zwykle — problemów należało ustalenie polskiej terminologii, która w tej dziedzinie nie jest dobrze ugruntowana. W różnych ośrodkach używa się różnych terminów na oznaczenie tego samego pojęcia, a zarazem niektóre terminy są niezbyt szczęśliwie dobrane. Ostatecznie terminologia *Słowniczka* stanowi wypadkową poglądów autora, który w przypadku nowych lub nieustalonych terminów starał się zachować już wprowadzone w zagranicznej literaturze fachowej lub firmowej, oraz poglądów redakcji, która walczy z zaśmiecaniem języka polskiego. Terminy równoważne (polskie lub angielskie) podano w nawiasach. Ponadto, ze względów praktycznych, jako hasła potraktowano również niektóre powszechnie stosowane skróty (ALU, ICE, RAM itp.).

ACKNOWLEDGE — p. potwierdzenie

ARBITRAŻ *(bus arbitration)* — procedura określania priorytetów jednostek zgłaszających żądanie dostępu do magistrali[1]) systemowej oraz przydzielania magistrali wybranej jednostce. Procedurę tę stosuje się najczęściej w przypadku systemów wielomikroprocesorowych, w których każdy mikroprocesor jest dołączony do tej samej magistrali sprzęgu wewnątrzkasetowego. Arbitraż jest realizowany za pomocą dodatkowych układów (praca równoległa) albo szeregowo na zasadzie łańcuchowej (ang. *daisy chain*).

ALU *(Arithmetic-Logic-Unit,* jednostka arytmetyczno-logiczna) — podstawowy blok mikroprocesora służący do wykonywania operacji arytmetycznych i logicznych.

BREAKPOINT — p. punkt wstrzymania

CCD *(Charge Coupled Device,* przyrząd o sprzężeniu ładunkowym) — w węższym znaczeniu: półprzewodnikowa pamięć o odczycie szeregowym, w której informacje są reprezentowane przez ładunek. Typowy średni czas dostępu pamięci CCD jest równy 15—100 μs przy częstotliwości przesuwu szeregowego równej 2—10 MHz.

CYKL MASZYNOWY *(machine cycle)* — wydzielona część cyklu rozkazowego, w czasie której odbywa się adresowanie, przesyłanie danych lub właściwa realizacja rozkazu. Jeden lub kilka cykli maszynowych składa się na jeden cykl rozkazowy, a — przykładowo — mikroprocesor INTEL 8080 ma 10 różnych cykli maszynowych.

CZAS DOSTĘPU *(access time)* — parametr pamięci, określający opóźnienie w pojawieniu się właściwych danych wyjściowych w stosunku do (nowego) adresu, przy ustalonej sekwencji sygnałów sterujących. Typowy czas dostępu pamięci półprzewodnikowych MOS o dostępie swobodnym równa się 250—450 ns, aczkolwiek dla najnowszych pamięci HMOS wynosi on zaledwie 15—25 ns.

DEBUGGING — p. uruchamianie

DMA *(Direct Memory Access,* bezpośredni dostęp do pamięci) — technika przesyłania danych, mająca na celu maksymalne zwiększenie szybkości transmisji między pamięcią a urządzeniem zewnętrznym — z pominięciem (mikro)procesora, który wtedy zwalnia szyny systemowe, tzn. przestaje wysyłać na nie sygnały.

DYSK ELASTYCZNY (dyskietka, *flexible disk,* niepopr. *floppy disk)* — elastyczny magnetyczny krążek, zamknięty w ochronnej kopercie, stanowiący nośnik do magazynowania danych. Dysk elastyczny ma zwykle średnicę 8 cali (203,2 mm) i pojemność 2—4 M bitów, a odmiana „mini" — średnicę 5 cali (127 mm) i pojemność 0,5—1 M bita. EAROM (Electricaly Alterable ROM) — pamięć zbliżona funkcjonalnie do EPROM, której zawartość może być elektrycznie zmieniana (aczkolwiek dużo wolniej niż odczytywana) bez usuwania układu z sytemu, tzn. na pakiecie montażowym. EAROM umożliwia przy tym selektywną zmianę danych — bez wymazywania całej zawartości.

EPROM *(Erasable PROM)* — pamięć PROM, której zawartość może być wymazana przez naświetlanie promieniami ultrafioletowymi, zazwyczaj w czasie kilku do kilkunastu minut.

FETCH — p. pobranie

FIRMWARE — p. oprogramowanie układowe

HANDSHAKING — sposób nawiązywania współpracy między dwoma urządzeniami pracującymi asynchronicznie, polegający na przesłaniu sygnału gotowości do wymiany danych i w odpowiedzi — potwierdzenia. Przykładowo — źródło danych informuje odbiornik o obecności danych wyjściowych, przy czym trwa to tak długo, aż odbiornik poinformuje zwrotnie o przyjęciu. *Handshaking* nie jest potrzebny wtedy, gdy odbiornik jest zawsze gotowy do pracy.

HOST COMPUTER — p. komputer macierzysty

ICE *(In-Circuit Emulator,* emulator układowy) — narzędzie sprzętowo-programowe umożliwiające badanie systemu mikrokomputerowego, a zwłaszcza uruchamianie oprogramowania w warunkach sprzętowych maksymalnie zbliżonych do rzeczywistych, przez dołączenie tego systemu do systemu wspomagania. Dołączenie polega na zastąpieniu mikroprocesora badanego systemu przez wtyk kabla łączącego.

INTERFACE — p. sprzęg

I/O — p. wejściowo-wyjściowy

KOMPUTER MACIERZYSTY *(host computer)* — w węższym znaczeniu: centralny komputer, w którym wykonuje się oprogramowanie skrośne.

[1]) Terminy zaznaczone rozstrzelonym drukiem są wyjaśnione w *Słowniczku* (przyp. red.)

KOMPUTER OSOBISTY *(personal computer)* — mikrokomputer, którego oprogramowanie — najczęściej zawarte w ROM — jest dostosowane do użytkowania przez niespecjalistów z dziedziny informatyki. Charakteryzuje się stosunkowo niską ceną dzięki temu, że jako pamięć zewnętrzną stosuje się najczęściej typowy magnetofon kasetowy, a telewizor — z reguły kolorowy — jako monitor ekranowy.

KURSOR — symbol (np. kreska, kwadrat lub gwiazdka) zwykle migający na ekranie monitora, służący do identyfikacji odpowiedniego wycinka pola ekranu odpowiadającego np. znakowi alfanumerycznemu albo elementarnemu obrazowi. Stosowanie kursora umożliwia bezpośrednie oddziaływanie operatora na ten wycinek pola.

MAGISTRALA — zestaw systemowych szyn danych, szyn adresowych oraz kontrolno-sterujących, a także szyn zasilania. Pojęcie stosowane najczęściej w kontekście sprzęgu wewnątrzkasetowego, tzn. na poziomie wkładanych do kasety pakietów.

MASKA — 1) słowo maszynowe, któreego bity służą do wyodrębnienia bitów innego słowa w celu dalszego ich wykorzystywania. Przykładowo maską może być: a) argument odpowiedniego rozkazu mikroprocesora, który „zasłania" niektóre bity słowa zawartego w akumulatorze; b) słowo kontrolne sterownika przerwań, które powoduje indywidualne „wycinanie" odpowiednich sygnałów wejściowych; 2) płytka szklana z obrazem (fotograficznym lub chromowym) topografii przyrządu półprzewodnikowego lub jego części.

MIKROKOMPUTER — mikroprocesor wraz z pamięciami (z reguły RAM i ROM) i układami pomocniczymi, takimi jak sterowniki oraz układy wejściowo-wyjściowe, a także z oprogramowaniem podstawowym zawartym w ROM, tzn. oprogramowaniu układowym.

MIKROKOMPUTER JEDNOUKŁADOWY *(single-chip microcomputer)* — mikrokomputer stanowiący pojedynczy (samodzielny) układ scalony, który zawiera zarówno mikroprocesor, jak i pamięć roboczą RAM oraz pamięć typu ROM, a także pewne układy wejściowo-wyjściowe. Przykładem mikrokomputera jednoukładowego jest mikrokomputer INTEL 8048 oraz INTEL 8051.

MIKROPROCESOR — centralna jednostka przetwarzania danych (CPU, *Central Processing Unit)*, zawierająca m.in. ALU, rejestry robocze i układy kontrolno-sterujące — wykonana z układów o dużym stopniu scalenia (LSI, względnie VLSI), albo też będąca pojedynczym układem, a nawet jego częścią.

MIKROPROCESOR SEGMENTOWY *(bit-slice microprocessor)* — mikroprocesor, którego część przetwarzająca dane jest tworzona z układów stanowiących 2- lub 4-bitowe segmenty. Architektura mikroprocesorów segmentowych umożliwia projektowanie mikroprocesorów o dowolnej, w stosunkowo szerokim zakresie, długości słowa oraz o specjalizowanych listach rozkazów, tworzonych na zasadzie mikroprogramowania. Mikroprocesory segmentowe są budowane najczęściej z układów bipolarnych, które w porównaniu z przeciętnymi układami LSI, wykonanymi technologią MOS, są szybsze, mimo mniejszego stopnia scalenia (por. sekwenser).

MULTIBUS — standardowy sprzęg wewnątrzkasetowy, tzn. zdefiniowany na poziomie modułów (pakietów) dołączonych równolegle do szyn systemowych (magistrali), wprowadzony pierwotnie przez firmę INTEL. Charakteryzuje się — poza sygnałami kontrolno-sterującymi — 20 liniami adresowymi, 16 dwukierunkowymi liniami danych, a także możliwością realizowania prostych systemów wieloprocesorowych.

NIBBLE (dosł. z ang.: ogryzek) — słowo 4-bitowe, również część bajtu odpowiadająca np. cyfrze w kodzie BCD.

OPROGRAMOWANIE SKROŚNE *(cross software)* — oprogramowanie podstawowe mające charakter narzędziowy (jak np. asembler, kompilator, symulator), umożliwiające tłumaczenie oraz uruchamianie programów (mikro)komputera innego niż (mikro)komputer macierzysty *(host computer)*, w którym oprogramowanie skrośne jest wykonywane.

OPROGRAMOWANIE UKŁADOWE *(firmware)* — 1) mikroprogramy umieszczone w ROM, 2) inne ustalone „firmowe" oprogramowanie podstawowe, jak np. edytor tekstowy i asembler, umieszczone w ROM.

MAREK TADEUSZ JANKOWSKI

KSIĄŻKI NADESŁANE

WYDAWNICTWA NAUKOWO-TECHNICZNE

Wacław Iszkowski, Marek Maniecki: Programowanie współbieżne. Nakład 4000 egz., 286 str., cena 110 zł, seria BIO, Warszawa 1982.

Książka zawiera omówienie podstawowych konstrukcji z dziedziny programowania współbieżnego, jak współpraca procesów i mechanizmy synchronizacji. Przedstawione są elementy najpopularniejszych języków programowania współbieżnego (CONCURRENT PASCAL, MODULA). Ponadto autorzy opisują zagadnienia związane z równoległością na poziomie języka wewnętrznego maszyny oraz dwa matematyczne modele dla systemów procesów: sieci Petriego i model automatowy.

Janusz M. Kwiek: Sterowanie produkcją zmiennoseryjną. Nakład 3000 egz., 142 str., cena 45 zł, Warszawa 1982.

Książka poświęcona jest metodyce projektowania systemów komputerowych służących sterowaniu produkcją. Autor — wychodząc z założenia, iż efektywne zastosowanie maszyn cyfrowych wymaga dobrego zrozumienia funkcjonowania organizacji, której mają one służyć — koncentruje się na dyskusji problemów z pogranicza informatyki i organizacji produkcji seryjnej wyrobów złożonych.

Witold Lipski: Kombinatoryka dla programistów. Nakład 5000 egz., 187 str., cena 120 zł, seria BIO, Warszawa 1982.

Recenzję tej książki zamieściliśmy w numerze 7/1982 naszego pisma.

George W. Struble: Wprowadzenie do programowania w języku asemblera dla maszyn cyfrowych IBM serii 360 i 370. Nakład 6000 egz., 502 str., cena 300 zł, Warszawa 1982.

Książka łączy naukę metod programowania w języku wewnętrznym maszyny z omówieniem komputerów IBM serii 360 i 370. Przedstawiono w niej budowę rozkazów, tryb adresowania, definiowanie i stosowanie makroinstrukcji, metody organizacji zbiorów danych, system przerwań, a także omówiono zasady uruchamiania programów.

Maksymilian Dryja, Janina i Michał Jankowscy: Przegląd metod i algorytmów numerycznych, część 2. Nakład 8000 egz., 276 str., cena 100 zł, seria BIO, Warszawa 1982.

Druga część monografii poświęconej metodom numerycznym zawiera omówienie rozwiązywania równań nieliniowych, układów równań liniowych, liniowego zadania najmniejszych kwadratów oraz równań różniczkowych cząstkowych, a także zadania wyznaczania wartości i wektorów własnych. Przedstawione są również własności numeryczne (poprawność, stabilność) omawianych algorytmów oraz ich złożoność obliczeniowa.

Romuald Jagielski: Tablice rozproszone. Nakład 5000 egz., 116 str., cena 40 zł, seria BIO, Warszawa 1982.

Książka została poświęcona pewnej klasie metod związanych z wyszukiwaniem informacji, a mianowicie metodom obsługi tablic rozproszonych (ang. hash tables). Omawiane podejście charakteryzuje się zaskakująco krótkim średnim czasem dostępu do poszukiwanych elementów. Autor przedstawia algorytmy tworzenia i modyfikacji tablic rozproszonych oraz ich zastosowanie. Ponadto czytelnik znajdzie w książce opis realizacji kartotek rozproszonych w języku PL/I.

Dennie Van Tassel: Praktyka programowania. Wydanie drugie, nakład 8000 egz., 297 str., cena 160 zł, seria BIO, Warszawa 1982.

Autor koncentruje się na takich zagadnieniach metodologicznych, jak: styl programowania, projektowanie programu, jego uruchamianie, testowanie i sprawność. W porównaniu z pierwszym wydaniem (WNT 1978) zmieniony został całkowicie rozdział o projektowaniu programów, znacznie też zmodyfikowano rozdziały o stylu programowania i o testowaniu. Książka stanowi próbę podsumowania dotychczasowych doświadczeń dających się ująć jako „umiejętność programowania".

Marek Głowacki: Systemy operacyjne DOS i OS. Nakład 8000 egz., 151 str., cena 60 zł, seria BIO, Warszawa 1982.

Książka omawia dwa najbardziej rozpowszechnione na świecie systemy operacyjne opracowane przez koncern IBM dla modeli komputerów z serii 360. Kolejne rozdziały zawierają opis sprzętu, organizacji zbiorów danych oraz zasad uruchamiania zadań i korzystania z zasobów maszyny. Na poziomie szczegółowości przyjętym przez autora, systemy DOS i OS właściwie nie różnią się od DOS/JS oraz OS/JS, zatem książka może być przydatna dla użytkowników maszyn Jednolitego Systemu.

Lance J. Hoffman: Poufność w systemach informatycznych. Nakład 5000 egz., 216 str., cena 120 zł, seria BIO, Warszawa 1982.

Autor dyskutuje o problemach związanych z ochroną danych we współczesnych systemach komputerowych. Prezentuje programowe i maszynowe sposoby zapobiegania nadużyciom w zbiorach danych, koncentrując się głównie na szyfrowaniu oraz na bezpieczeństwie programów systemowych. Oprócz przeglądu standardowych metod, czytelnik znajdzie omówienie architektury maszyn i jej wpływu na bezpieczeństwo danych oraz przegląd obszarów przyszłych badań.

GRY

Polowanie na UFO

Począwszy od niniejszego numeru prezentować będziemy na naszych łamach pomysły gier komputerowych. Stanowią one doskonałą zabawę dla informatyków, a jednocześnie pomagają w przełamaniu strachu przed maszyną cyfrową, odczuwanego przez „niewtajemniczonych". Zaprogramowanie jest na ogół sprawą prostą, natomiast wymyślenie oryginalnych i interesujących zasad nastręcza znacznie więcej problemów. Dlatego też na łamach INFORMATYKI przedstawiać będziemy wyłącznie pomysły, natomiast teksty zapisane w FORTRANIE możemy wysyłać na życzenie.

Zachęcamy wszystkich Czytelników do nadsyłania własnych (i nie tylko) pomysłów gier. Poniżej przedstawiamy pierwszą z nich.

Kosmici w latającym talerzu dokonują inwazji na Ziemię. Jesteś jednym z obrońców. Przestrzeń nad Tobą podzielona jest na sześciany o wymiarach 1 km × 1 km × 1 km. Masz do dyspozycji ruchomą wyrzutnię rakietową typu ziemia-powietrze. Rakieta wystrzelona w pewien sześcian niszczy jego zawartość. Niestety, UFO wytworzył sztuczną mgłę i posiada urządzenia zakłócające namiar kierunkowy, tak że w każdym momencie możesz określić tylko jego odległość od Twojej pozycji. Latający talerz stoi w miejscu do czasu oddania przez Ciebie strzału, bądź zbytniego zbliżenia się do jego pozycji. Jeśli UFO porusza się, to może oddać strzał w Twoim kierunku (niekoniecznie celny). Gra kończy się zniszczeniem jednego z przeciwników.

Proponujemy następujące ograniczenia:
— obszar walki 150 × 150 × 50 km
— zasięg urządzeń lokacyjnych 90 km
— zasięg rakiet 65 km.

Ponieważ opis gry pozwala na dużą dowolność interpretacji będziemy wdzięczni za nadsyłanie nam konkretnych implementacji.

Z. ŚWIRSKI

Ceny i monopol

Zawsze doskwierały nam ceny — za sprzęt, za szkolenie, projektowanie, programowanie. Jakiekolwiek porównanie z cenami światowymi — porównanie do średniej płacy, oczywiście — wykazywało, że producenci czegokolwiek w informatyce powinni opływać w niebotyczne dostatki. Było i jest ku zdumieniu każdego, kto starał się przyjrzeć życiu polskich firm informatycznych, niemal odwrotnie. Ani dostatku, ani zadowolenia. Był co najwyżej spokój bez wzruszeń. Nie było radości z sukcesów, ale nie było też niepokoju z poczucia zagrożenia.

Produkowanie żelaza lub — jeśli kto woli — hardwar'u może być, jak autorytatywnie głoszą biegli w tajnikach ekonomii, tanie i opłacalne dla producenta tylko przy wielkich seriach. Na pewno mają rację; niekiedy tylko zadaję sobie pytanie — skoro nie możemy produkować wielkich serii w dużych fabrykach, dlaczego nie produkujemy tychże komputerów w małych zakładach (zakładzikach może?); przecież ich produkcja jest i tak j e d n o s t k o w a. Jeszcze w latach sześćdziesiątych ktoś badał sprawę zorganizowania takiej produkcji — przeszkodziła wówczas doktryna. Może nie doktryna, a pogląd urzędnika powołującego się na doktrynę. Jaką!?

Świat obecnie oszalał na punkcie informatyki, mikrokomputery zaglądają niemal wszędzie, na ich oprogramowaniu robi się ogromne interesy. Robiło się je również i dawniej — już bowiem od mniej więcej początku lat siedemdziesiątych ciężar rynku przeniósł się właśnie na oprogramowanie i jego pielęgnację. Jak grzyby po deszczu wyrastają „software hause'y" duże, średnie lub wręcz całkiem małe — kilkuosobowe, oparte na jednym pomyśle, na jednym typie programów.

Czemu u nas ten typ działalności nie może nabrać rozmachu? Przyczyn jest wiele. Jedna z nich ma — moim zdaniem — głębokie korzenie w powszechnie panującej zasadzie, którą niektórzy nazywają polskim piekłem; zasadzie, według której trzeba najpierw wytropić, zniszczyć i wszelkimi sposobami zabezpieczyć się przed nadużyciem, a dopiero później można robić cokolwiek konstruktywnego. Trzeba wszystko znormować, poukładać, ponazywać, wycenić, najlepiej wg ton, kilometrów, godzin. Do wymierzenia oprogramowania ani tony, ani metry nijak się nie nadają, wobec tego przyjęto godziny i płaci się za nie, nie zaś za pomysł czy staranność.

Mierzenie wydajności, normowanie ma oczywiście sens, choćby taki, że można dzięki niemu cokolwiek przewidzieć i zaplanować. Ma sens jako wynik szerokich obserwacji statystycznych. Myli się jednak ten, kto chce w oparciu o nie zaplanować, a później dokładnie rozliczyć konkretne przedsięwzięcie, w którym decydującą rolę, tak jak w programowaniu, odgrywa czynnik ludzki.

Tu mała dygresja. Z badań przeprowadzonych przez IBM jeszcze w połowie lat siedemdziesiątych wynikało, że wydajność programistów waha się w granicach od 1 do 30. Przez wydajność rozumie się nie tylko szybkość uruchamiania programów, ale ich efektywność (zwartość kodu, czas wykonywania programu itp.), a więc wszystko to, co ma bezpośredni wpływ na poprawność i użyteczność programu.

Zastanówmy się, do czego prowadziły (i jeszcze długo będą prowadzić — poprzez przyzwyczajenia „fachowców" wychowanych w tym stylu) normy i cenniki godzinowe.

Po pierwsze — nie ma chyba w Polsce ani jednego zespołu programistów, który byłby więcej niż średni. Średni to znaczy c o ś k i e d y ś może i z r o b i ł, ale się nie przemęcza, nie kipi pomysłami, nie śledzi literatury, nie szuka nowych pomysłów. Bo po co? Wystarczy jaka taka niezła marka i umiejętność planowania nadmiarowego. Oszczędności czasu nic nie dają; nawet genialny pomysł, jeśli powstał za szybko i szybko został zrealizowany, jest mało wart. Najlepsze są pomysły, z których — jak to się mówi — i wnuk się pożywi — będzie jeszcze projektował, wdrażał, projektował...

Co wobec tego daje satysfakcję, jeśli nie tworzenie i sukces na rynku. Ano pozostaje jedynie potęga firmy i hierarchia. Stąd i wielkie kolosy (których nawet IBM mógłby pozazdrościć) — z armią kierowników, głównych i główniejszych specjalistów. Krzątanina przy planach, planszach, reorganizacjach i naradach, a poza tym — marazm. Niewielkie ożywienie zdarza się tylko wtedy, gdy zbliża się termin odbioru, przesunięty do możliwych najdalszych granic. Po co się spieszyć — programowanie, poza spokojnym egzystowaniem, niczego przynieść nie może.

Doprawdy zadziwiające, że jednak polscy programiści biorą skądś wiedzę i umiejętności (może z nudów czegoś się uczą, o czymś myślą) oraz sukcesy — gdy zaczepią się gdzieś za granicą, gdzie liczy się wynik (tym bardziej, im szybciej i lepiej osiągnięty).

Główną przyczynę opisywanego stanu zilustruje zasłyszana dykteryjka. Brzmi ona nieco surrealistycznie, ale jest autentyczna — Anno Domini 1982.

Jeden z informatyków, pewien swych umiejętności, uwierzył, że mamy już reformę, a ponieważ chciał się przyczynić do hamowania inflacji — postanowił więc rozejrzeć się za możliwością zorganizowania małej, malutkiej spółdzielni programistów. Dyskusje i wyliczenia upewniły go, że nawet na obecnym rynku informatycznym minispółdzielnia rozkręci się i to nie tylko z pożytkiem dla jej członków.

Wybrał się był (w lipcu, może we wrześniu), do bardzo Wysokiego Urzędu (wyższego już chyba nie ma) z pytaniem, czy Urząd byłby skłonny poprzeć pomysł i co w ogóle o tym myśli. Sądził biedak, że rozstrzygającym argumentem, po którym natychmiast go uściskają, dając odpowiednie papiery i błogosławieństwo, będzie to, że spółdzielenka może stosować ceny pięciokrotnie niższe niż wspaniałych „Domów Programowania", terminy na pewno dwa razy krótsze, a i solidność wielka (założyciele przewidzieli wiele kruczków i zameczków, by miernoty programistycznej do biznesu nie dopuścić).

Przyjęto go uprzejmie i, owszem, wysłuchano z początku nawet uważnie, ale niestety pomysłu nie pochwalono — co by bowiem zrobiły „Domy Programowania"? To przecież konkurencja! Naiwny skwapliwie potwierdził, że właśnie o to chodzi, że to da uśpionym mamutom bodziec do lepszego działania i ożywienia. Tak nie można, tam są przecież ludzie, nie można im stwarzać zagrożenia — odpowiedzieli urzędnicy. Jedyne co radzili, to opracowanie memoriału o tym, jak należy poprawić owe „Domy", memoriału, który gdy trafi gdzie trzeba — najlepiej na biurko generała — to natychmiast on coś poradzi — wyśle kogo trzeba i wszystko będzie dobrze.

Niestety, historyjka jest prawdziwa. Spółdzielnia nie powstała. A wszystko kręci się starym, sprawdzonym trybem.

JANUSZ GWIAZDA

Cena zł 75.—

W numerze 2/83 przedstawimy dalszą część
TECHNIKI MIKROPROCESOROWEJ

Grzywak A., Suchorończak Z.: Rozwój systemów mikrokomputerowych na przykładzie MERY-60 (SM--1633)

Pieńkos J. i in.: Modułowy system sterowników mikroprocesorowych MIKROSTER

Woźniak A. i in.: Modułowy System Mikroprocesorowy — MSM

Jagiełło S., Koźmiński A., Rzymkowski K.: Wybrane mikrokomputerowe bloki sterujące systemu CAMAC i mikrokomputery o przeznaczeniu uniwersalnym

Binkowski W., Przybylski A., Furyk M.: Alfaskop ALFA 31M jako urządzenie wejścia-wyjścia systemu mikrokomputerowego

Jankowski M.T., Wertel J.: Jednopakietowy inteligentny sterownik graficzny MSG-3C

W najbliższych numerach:

- Przegląd mikroprocesorów 16-bitowych
- Dyskowy system operacyjny CP/M
- Systemy graficzne o organizacji rastrowej
- Raport dla Klubu Rzymskiego: Mikroelektronika i społeczeństwo. Na dobre i złe (fragmenty)